ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

FACULTAD INFORMATICA Y ELECTRONICA

ESCUELA INGENIERIA EN SISTEMAS

TEMA:

"ANÁLISIS DE FRAMEWORKS DE PRESENTACIÓN PARA EL DESARROLLO DE APLICACIONES WEB EN JAVA, CASO PRÁCTICO: GADPCH".

TESIS DE GRADO

Previa a la obtención del título de:

INGENIEROS EN SISTEMAS INFORMÁTICOS

Presentado por:

Mercedes Elena Morán Tapia

Ximena Fernanda Moposita Saltos

RIOBAMBA-ECUADOR

2012

AGRADECIMIENTO

Dejamos constancia de nuestro profundo agradecimiento:

A la ESPOCH y a todos los docentes que nos brindaron sus conocimientos necesarios para lograr obtener una educación de calidad.

A nuestro director de tesis el Ing. Julio Santillán y asesor Ing. Ing. Danny Velazco quienes nos guiaron con su sabiduría y consejos prestándonos su tiempo, dedicación y confianza a la hora de desarrollar la presente investigación.

Al Gobierno Autónomo Descentralizado de la Provincia de Chimborazo por habernos dado la oportunidad de desarrollar nuestro trabajo de investigación, en especial al Ing. Jaime Zárate que nos brindo ayuda para aplicar nuestros conocimientos en dicha institución.

DEDICATORIA

La presente tesis la dedico con mucho amor y cariño:

A mi Dios por haberme dado la oportunidad de vivir, saber guiar mi camino y cuidarme cada día de mi vida.

A mis padres Jorge y Mélida por ayudarme y brindarme su amor incondicional para llegar a alcanzar mí meta.

A mi adorado hijo Jair que es el pilar fundamental en mi vida y la razón para seguir adelante en busca de un mejor futuro.

A mis hermanos Marcelo, Marisol por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida. Y a Johana por ser el ejemplo de hermana mayor y una amiga incondicional gracias por siempre confiar en mí, apoyarme y enseñarme a levantarme luego de una caída.

Ximena

Dedico mi tesis primero a Dios y a la Virgen María por derramar infinitas bendiciones, acompañándome en los buenos y malos momentos, dándome vida, sabiduría, paciencia y permitiéndome alcanzar mis objetivos.

A mis padres José y Cristina por estar siempre en cada etapa de mi vida dándome su amor incondicional, su apoyo constante y fuerza para continuar cada día.

A mis hermanas Marcia y Juanita por ser mis ejemplos de superación, enseñándome alcanzar mis metas hasta el fin y siendo siempre unas amigas incondicionales.

A mi cuñado Wilmer por ser como mi hermano y a mi sobrina Angie que es mi alegría.

Mercedes

FIRMAS DE RESPONSABLES Y NOTA

NOMBRES	**FIRMAS**	**FECHA**
Ing. Iván Menes	……..	……..
DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA		
Ing. Raúl Rosero	……..	……..
DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS		
Ing. Julio Santillán	……..	……..
DIRECTOR DE TESIS		
Ing. Danny Velazco	……..	……..
MIEMBRO DE TESIS		
Tlgo. Carlos Rodríguez	……..	……..
Dir. Dpto. CENTRO DOCUMENTACIÓN		
NOTA DE LA TESIS	……..	……..

RESPONSABILIDAD DE LAS AUTORAS

Nosotras, “Mercedes Elena Moran Tapia y Ximena Fernanda Moposita Saltos, somos las responsables de las ideas, doctrinas y resultados expuestos en esta Tesis de Grado, y el patrimonio intelectual de la misma pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”.

FIRMAS:

Mercedes Elena Morán Tapia	Ximena Fernanda Moposita Saltos

PORTADA

AGRADECIMIENTO

DEDICATORIA

ÍNDICE GENERAL

CAPÍTULO I

CAPÍTULO II

ÍNDICE DE ABREVIATURAS

API: Application Programming Interface.

HTTP: Hypertext Transfer Protocol

MVC: Modelo, Vista, Controlador.

SOAP: Simple Object Access Protocol

XML: Extensible Markup Language.

HTML: Hipertext Transfer Protocol

IDE: Entorno de Desarrollo Integrado.

J2EE: Java 2 Enterprise Edition

JDK: Java Development Kit

JSP: Java Server Pages.

UI: Interfaz de Usuario.

CSS: Cascading Style Sheets.

EJB: Enterprise Java Beans

BD: Base de datos.

MYSQL: My Structured Query Language

JDBC*:* Java DataBase Conectity.

JSF: Java Server Faces.

POJO: Plain Old Java Object

OGNL: Object-Graph Navigation Language

CGI: Common Gateway Interface

URL: Uniform Resource Locato

ORM: Object Relational Mapping

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el desarrollo de Aplicaciones Web de forma constante se evoluciona y con este llegan mejoras e innovaciones en el entorno de programación permitiendo que en la construcción y diseño se disminuya el tiempo de desarrollo.

Java es un recurso inestimable que posee una buenas perspectivas del futuro para el desarrollo de Aplicaciones Web además es muy famoso ya que es un lenguaje independiente de la plataforma. Por esta razón se ha optado por hacer un análisis comparativo de frameworks de presentación en Java y aplicarlo el mejor en la realización de la Aplicación Web.

Con un análisis previo de frameworks de presentación (Struts2, JSF, Cocoon, Tapestry y Spring MVC) se pudo seleccionar Strust2 y Spring MVC para realizar un análisis comparativo más profundo que ayudo a desarrollar la Aplicación Web de la Unidad de Archivo General del Gobierno Autónomo Descentralizado de la Provincia de Chimborazo con mayor rapidez.

La estructura de este documento se encuentra conformada por cuatro capítulos los cuales se detallan a continuación:

El **capítulo I**, contempla una descripción de las razones fundamentales sobre el desarrollo de Aplicaciones Web en entornos Java y que objetivos permitirá alcanzar.

En segunda instancia se tiene el **capítulo II** que da una referencia teórica sobre conceptos esenciales de los frameworks de presentación en Java para aplicaciones Web.

El **capítulo III** contiene el análisis previo de los frameworks de desarrollo de Aplicaciones Web (Struts2, JSF, Cocoon, Tapestry y Spring MVC) y el análisis comparativo más profundo (Struts2 y Spring MVC) para determinar el más adecuado. Finalizando con la comprobación de la hipótesis de la investigación realizada.

Y por último el capítulo **IV** que contiene en detalle la creación de la Aplicación para la Gestión de Archivos (SARGE) del Gobierno Autónomo Descentralizado de la Provincia de Chimborazo.

CAPÍTULO I

MARCO REFERENCIAL

1.1. Antecedentes

Con la constante evolución y crecimiento del internet causa un gran auge el desarrollo de aplicaciones Web permitiendo compartir y gestionar información de forma transparente y desde cualquier lugar.

Dentro del desarrollo de este tipo de aplicaciones cada día se mejora e innova las herramientas para que la construcción y diseño de un entorno Web sea más fácil y rápido de desarrollar. Java juega un papel muy importante actualmente, ya que representa uno de los mejores medios para construir dichas aplicaciones, su falta de sencillez ha provocado la aparición de diferentes frameworks que facilitan las tareas a los desarrolladores. Los frameworks en si son un conjunto de componentes con un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web, permitiendo a los diseñadores y programadores

concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así facilita ciertas funcionalidades básicas y comunes.

Para la capa de presentación se quiere el framework en Java que proporcione mayor facilidad en la elaboración de pantallas, mapeo entre los formularios y sus clases en el servidor, la validación, conversión, gestión de errores, traducción a diferentes idiomas y, a ser posible, que facilite la inclusión de componentes complejos (menús, árboles, Ajax, etc) de forma sencilla y, sobre todo, fácil de mantener.

Debido que en el mercado existe una gran cantidad de frameworks para la capa de presentación para el desarrollo de aplicaciones Web en Java, lo que podría convertirse en un problema poniendo al programador a realizar un análisis muy riguroso a las fortalezas y debilidades entre las diferentes alternativas antes de elegir aquel que mejor se ajuste a las necesidades del proyecto. Cada framework tiene su propio origen y filosofía para alcanza el mismo fin, por esta razón, no es tan fácil seleccionar la mejor opción al momento de desarrollar una aplicación Web. Entre las tesis investigadas no se encontró ninguna que muestra un análisis de los frameworks con este enfoque que se muestra en este proyecto de investigación.

El Gobierno Autónomo Descentralizado de la Provincia de Chimborazo (GADPCH) requiere un sistema web para el control de archivos, en la unidad de Archivo General.
El mismo que consiste en que los documentos (entendiéndose por oficios, proyectos, leyes, etc.) que provienen de las distintas coordinaciones y de sus divisiones (unidades y

secciones), son enviados a la Unidad de Información que los organizan en cajas que posteriormente llegan a la Unidad de Archivo General donde se organizan nuevamente para ser almacenados en carpetas que a su vez estarán dentro de una caja, bandeja, estantería, depósito y pertenecerá a una sección de bodega.

Estas personas encargadas llevan actualmente la organización y ubicación de los documentos en diez matrices para lo cual se utilizan en hojas de papel y también información que se almacena en Excel 2007, principalmente para manejar lo que es un Inventario de Documentos de Archivos, una Tabla de Plazos de Conservación Documental y Préstamos de Documentos tanto para funcionarios del GADPCH o personas particulares.

1.2. Justificación.

1.2.1. Justificación teórica.

Una de las capas más importante es la capa de presentación ya que encapsula toda la lógica para dar servicio a los clientes que acceden al sistema. Esta capa intercepta las peticiones de cliente, provee capacidades de login, mantiene la sesión del cliente, controla el acceso a los servicios de negocio y construye respuestas que devuelven al cliente. J2EE no incluye ninguna utilidad, librería, API o mecanismo que proporcione esta funcionalidad, sino que permite la libertad y flexibilidad de diseñar e implementar cualquier solución mediante los mecanismos que proporciona: Servlets, JSPs, custom tags, etc.

Si no se realiza un buen diseño en esta capa, una aplicación web puede llegar a volverse compleja y difícil de mantener. Por este motivo, se ve la importancia de utilizar frameworks con el objetivo de facilitar la creación de diseños Web exitosos así como reducir la complejidad.

No es obligatorio utilizar un framework para la construcción de aplicaciones Web, pero estos son de mucha importancia para la productividad y mejora en los desarrollos, permitiendo mayor simplificación, ahorrando tiempo y trabajo a los desarrollos colaborativos.

Actualmente existe una gran cantidad de frameworks de presentación disponibles para facilitar el desarrollo de aplicaciones web. El abanico de opciones es muy amplio entre los que se encuentra Struts, Spring, JSF, Tapestry, Cocoon y por ese motivo, antes de comenzar a utilizar uno de ellos se necesita evaluarlos muy detenidamente ya que su complejidad puede afectar positiva o negativamente al tiempo de desarrollo de la aplicación.

Dicha evaluación se centra en tener una idea clara de cada uno de los frameworks, dando una visión completa de los aspectos más relevantes de cada uno de ellos, permitiendo seleccionar dos de ellos y analizar profundamente los diferentes enfoques y características que tienen y de esta manera facilitar la elección del framework que brinde mayores beneficios para desarrollar la aplicación Web.

1.2.2. Justificación práctica

La razón principal para el desarrollo de este proyecto surge de las necesidades que existen, debido a que estos documentos se manejan de manera manual y rudimentaria lo que con lleva un consumo excesivo de recursos económicos, materiales y de talentos humanos, y además en el tiempo en que se demoran en realizar dicha actividad que influye en el desarrollo de la misma.

Así que con la visión de optimizar y enlazar al máximo las nuevas capacidades implementadas es posible dar una solución informática creando un Sistema de Archivo General (SARGE) permitiendo a los usuarios contar con un ordenamiento alfanumérico y cronológico de la documentación que se requiere mantener en el Archivo General del GADPCH.

1.3. Objetivos.

1.3.1. Objetivo General.

- ✓ Realizar un análisis de frameworks de presentación para el desarrollo de aplicaciones Web en Java, para el Gobierno Autónomo Descentralizado de la Provincia de Chimborazo

1.3.2. Objetivos Específicos.

- ✓ Estudiar las características, ventajas y desventajas de los frameworks seleccionados de la capa de presentación para una aplicación Web en Java.
- ✓ Establecer parámetros de evaluación para elaborar la comparativa entre frameworks de presentación Web en Java.

- ✓ Evaluar los frameworks Web de la capa de presentación en Java.
- ✓ Desarrollar una aplicación Web de acuerdo con el framework de la capa de presentación más adecuada para la unidad de archivo del GADPCH.

1.4. Hipótesis.

Mediante el análisis entre frameworks de capa de presentación se permitirá seleccionar el más adecuado reduciendo el tiempo de desarrollo de aplicaciones Web.

CAPÍTULO II

FRAMEWORKS WEB PARA JAVA

2.1. Generalidades

2.1.1. J2EE

Actualmente, las empresas tienen un conjunto de necesidades bastante diferentes de las que tenían hace unos años. En la era de la información, toda empresa que quiera ser competitiva en su sector tiene que ofrecer sus servicios a sus clientes, empleados, *partners*, etc.

Hoy todas las entidades permiten a sus clientes operar por Internet, incluso hay entidades que sólo operan por Internet, ofreciendo sus servicios por el mismo.

Estos sistemas que los proporcionan tienen que cumplir una serie de características:

- ✓ **Alta disponibilidad**. Los servicios que se ofrecen no pueden dejar de funcionar. Se tienen que proporcionar mecanismos para asegurar que no dejen de funcionar y, si lo hacen, que el tiempo de interrupción sea mínimo.

- ✓ **Mantenibles.** Se tiene que poder añadir servicios a los sistemas y modificar los existentes fácilmente.
- ✓ **Seguros.** Se tiene que garantizar que no haya accesos no autorizados a los servicios y se deben establecer políticas de acceso para los diferentes tipos de usuarios de estos servicios.
- ✓ **Fiables**. Los servicios tienen que ser el máximo de fiables y libres de errores. Se debe ofrecer mecanismos de detección y diagnosis de errores.
- ✓ **Escalables**. Si aumenta la carga del sistema, tiene que ser fácil añadir servidores o bien ampliar los que ya se tiene para dar la misma calidad de servicio sin tener que modificar las aplicaciones existentes.
- ✓ **Portables**. Se tiene que poder cambiar de plataforma de una manera no traumática. Un cambio de plataforma no puede implicar volver a implementar todos los servicios.
- ✓ **Rápidos de desarrollar y de desplegar.** Se tiene que poder desarrollar y ofrecer servicios a los usuarios del sistema de manera ágil y rápida. Hoy en día, el famoso *time-to-market* es vital para las empresas.
- ✓ **Fácilmente integrables con los sistemas existentes.** Raramente se desarrollarán servicios nuevos partiendo de cero. Lo normal será integrar los nuevos servicios con servicios o desarrollos ya existentes, y se lo tiene que poder hacer fácilmente.

Estas características son comunes a todos los servicios que se desea ofrecer. La mejor forma de obtenerlas es utilizando un conjunto de servicios de bajo nivel que las

proporcionen. Éstos se pueden conseguir de dos maneras: utilizando una plataforma de desarrollo que facilite todos estos servicios o cargando estas tareas de bajo nivel y bastante complejas en los desarrollado-res de los servicios de negocio. Esta es la diferencia básica entre utilizar una plataforma de desarrollo empresarial y no hacerlo. En el segundo caso, los desarrolladores perderán mucho tiempo implementando servicios de bajo nivel que tienen poco que ver con los servicios de negocio que hay que ofrecer (y que son realmente el objetivo de la empresa).

Se necesita una plataforma que ofrezca un conjunto de servicios a los arquitectos y a los desarrolladores para facilitar el desarrollo de aplicaciones empresariales.

2.1.1.1. ¿Qué es J2EE?

J2EE es una plataforma de desarrollo empresarial propuesta por Sun Microsystems que define un estándar para el desarrollo de aplicaciones multicapa teniendo como objetivo primordial hacer más sencillo y ágil el desarrollo y la puesta en marcha de aplicaciones empresariales.

J2EE es otro acrónimo en el mundo de la informática. Este representa *Java 2 Platform, Enterprise Edition*. Su importancia se pondrá de manifiesto, una vez que traza su linaje.

La plataforma Java es una máquina virtual, una mirada procesador igual que traduce las instrucciones informatizada en funciones.

El lenguaje Java es tal que permite la comunicación entre la plataforma entre múltiples tipos de dispositivos. Por ejemplo, un programador puede desarrollar un código de Java en

un equipo de escritorio y esperar que se ejecuten en otros equipos, routers, e incluso los teléfonos móviles, siempre que esos dispositivos son compatibles con Java. Un gran número de computadoras centrales, computadoras, teléfonos móviles y otros dispositivos electrónicos funcionan con la plataforma Java. El *2* en la J2EE acrónimo de *versión 2*. Como ocurre con muchas aplicaciones de software, J2EE es la plataforma Java Versión 2. En realidad, el número 2 es a menudo cayó hoy en día, por lo que J2EE se convierte en Java EE. Tradicionalmente, sin embargo es todavía J2EE. [1]

2.1.1.2. Arquitectura J2EE.

Está basada completamente en un modelo de 4 niveles lógicos:

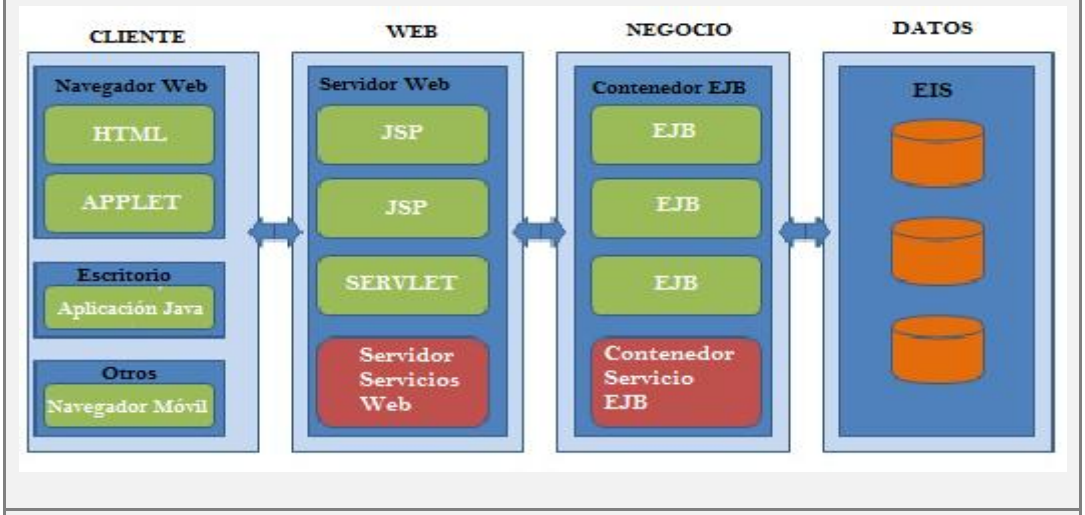


Figura II.1: Niveles de arquitectura J2EE

[1] *¿Qué es J2EE?*
http://lular.es/a/Internet/2012/01/Que-es-J2EE.html

- ✓ *Nivel cliente*, como su propio nombre indica, nivel al que pertenece el cliente de la aplicación. Es la interfaz gráfica del usuario e interactúa con el usuario. En J2EE se da soporte para distintos clientes (HTML, Applets y aplicaciones Java entre otros).
- ✓ *Nivel web*, nivel al que pertenece el contenedor web, encargado del manejo de elementos Servlets y JSP. Recibe las peticiones del cliente y en función de éstas se genera una respuesta adecuada.
- ✓ *Nivel negocio*, en éste se encuentra el servidor de aplicaciones, encargado de la lógica de negocio del sistema. Los elementos manejados por esta capa son los EJB (que a su vez interactúan con la capa de datos).
- ✓ *Nivel datos*, nivel en el que se enmarca el sistema de información empresarial (EIS) que comprende la base de datos, sistema de procesamiento de datos y sistemas legados.

2.1.1.3. Componentes, contenedores, servicios y servidores de aplicaciones de J2EE

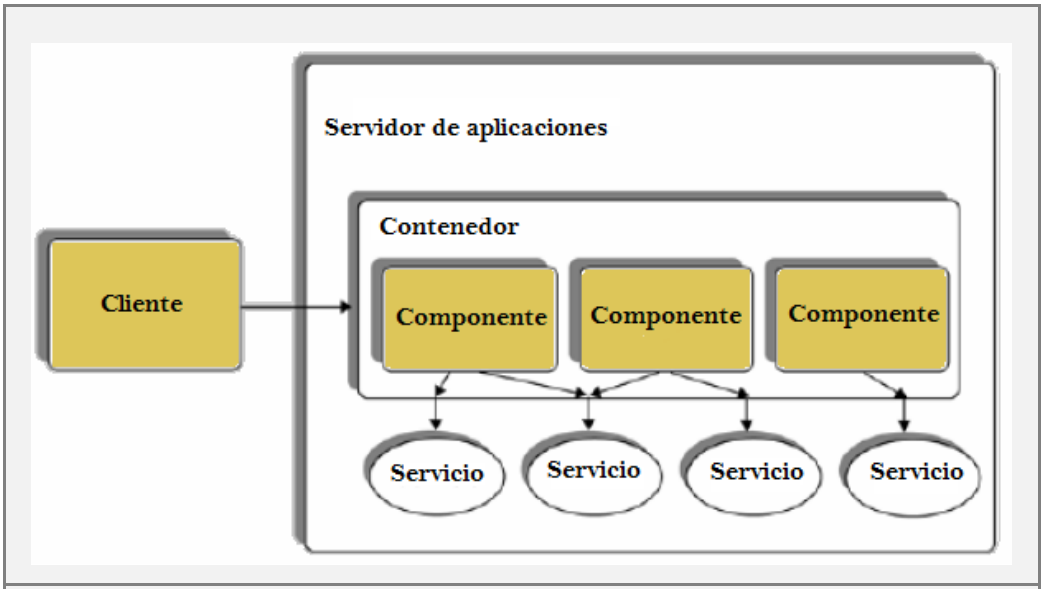


Figura II.2: Componentes, contenedores, servicios y servidores de aplicaciones

2.1.1.3.1. Componentes de J2EE

Un componente es la unidad básica, funcional e independiente de software que satisface una especificación de la plataforma. Se definen los principales componentes:

- ✓ ***Applets*.** Son componentes java que interactúan dentro de un navegador web y proporcionar una interfaz web para aplicaciones J2EE. Se ejecutan en un contenedor de applets.
- ✓ ***Componentes Web*.** Se ejecutan en un servidor web para responder a peticiones HTTP realizadas por clientes, generando código HTML o XML generalmente:
  - *Servlets.* Son clases Java que procesan requerimientos de un cliente y generan la acción pertinente, adecuada a la petición.
  - *JSP (Java Servlets Pages).* Permiten crear componentes de la capa de vista, utilizando código HTML y XML. Permiten realizar procesos e integrar datos con el fin de generar datos de forma dinámica compartidos con datos de plantillas estáticas.
  - *EJB (Enterprise Java Beans).* Representan la lógica de negocio de la aplicación. Se ejecutan en un ambiente distribuido y soportan transacciones. Encapsulan el acceso al EIS a través de la utilización de objetos que proveen la funcionalidad del manejo de transacciones y persistencia.

2.1.1.3.2. Contenedores J2EE

Los contenedores son servicios que ofrecen soporte para la ejecución de un componente y comunicación con otros homólogos. Existe un contenedor para cada tipo de componente:

- ✓ ***Contenedor de applets***. Proporciona un programa anfitrión en el cual se desenvuelve la ejecución del applet.
- ✓ ***Contenedor web***. Encargado de proporcionar las funciones de comunicación con clientes HTTP, generación dinámica de contenidos y presentación de datos.
- ✓ ***Contenedor de EJB***. Encargado de proporcionar servicios comunes a la lógica de negocio tales como la distribución de objetos, la persistencia, la concurrencia y las transacciones.

2.1.1.3.3. Servicios J2EE

J2EE aporta los siguientes servicios estándares, incluyendo las APIs (son las interfaces que proporciona un sistema de software como componente, servicio, aplicación para que los clientes puedan interactuar) necesarias para su implementación. Los más comunes son los que se citan a continuación:

- ✓ ***HTTP y HTTPS***. Protocolos estándares para comunicaciones web y comunicaciones seguras sobre Secure Socket Layer (SSL). La API cliente está definida por Java.net.* y la del servidor por las clases de Servlet y JSP.
- ✓ ***JDBC (Java DataBase Conectity).*** API para el acceso de bases de datos relacionales.
- ✓ ***JavaMail.*** API para la creación de aplicaciones de mensajería independientes de la plataforma y el protocolo.
- ✓ ***RMI-IIOP (Remote Method Invocation-Internet Inter ORB protocol).*** APIs que permiten la programación distribuida a través de RMI.

- ✓ ***JavaIDL (Java Interface Definition Languaje).*** API para la invocación de servicios CORBA.
- ✓ ***JTA (Java Transaction API).*** API que permite el manejo de transacciones (para inicialización, terminación o aborto de transacción).
- ✓ ***JNDI (Java Naming and Directory Interface).*** API para el registro y acceso de servicios y objetos.
- ✓ ***JAXP (Java API for XML Parsing).*** Realiza el procesamiento de ficheros XML a partir de las APIs SAX, DOM y XSLT.
- ✓ ***JCA (J2EE Connector Architecture).*** API cuyo cometido es la agregación de recursos nuevos a cualquier aplicación J2EE.
- ✓ ***JAAS (Java Authentication and Authorization).*** Permite la identificación de usuarios y autorización para acceder a recursos de la aplicación. [2]

2.1.1.3.4. Servidores de aplicaciones J2EE

Los servidores de aplicaciones J2EE son piezas de software que implementan los contenedores, servicios y API que define la plataforma J2EE.

Hacen de "cola" entre todos estos elementos, y proporcionan un entorno de gestión, despliegue y ejecución integrado para las aplicaciones desarrolladas bajo la plataforma J2EE. Estos servidores hacen que los componentes, contenedores y servicios estén sincronizados y trabajen conjuntamente para proporcionar una serie de funcionalidades.

---

[2] *Clases y características de J2EE*
http://es.scribd.com/doc/34022342/7/Clases-caracteristicas-de-J2EE

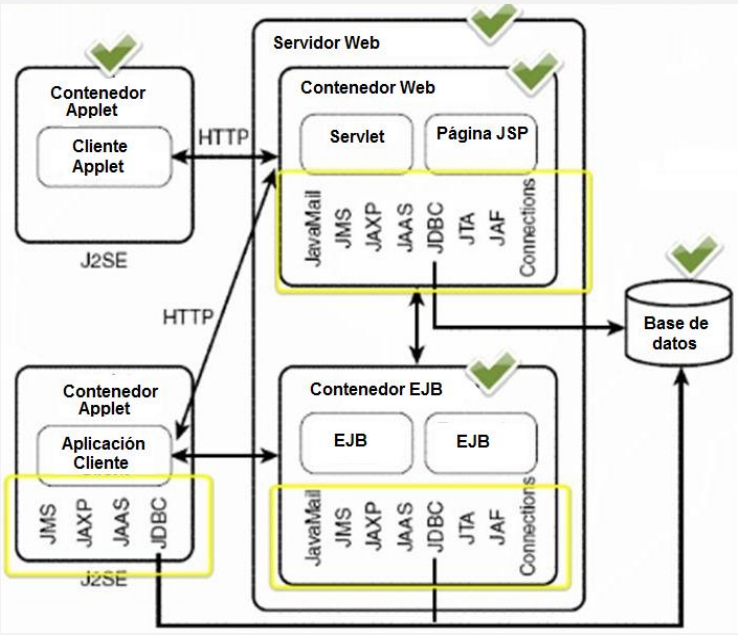


Figura II.3: Arquitectura de J2EE con sus servicios

Estos servidores residirán los componentes empresariales, bien sean objetos distribuidos accesibles remotamente, componentes web, páginas web o incluso applets. Es importante remarcar que cualquier aplicación J2EE se tiene que desplegar en un servidor de aplicaciones para que los clientes puedan interactuar con ella. Cualquier servidor de aplicaciones que implemente las especificaciones de la plataforma J2EE tiene que ofrecer todas las tecnologías, API y servicios que estén incluidos como obligatorios en la especificación.

Un servidor de aplicaciones J2EE tendrá normalmente un contenedor de servlets, un contenedor de EJB, un sistema de mensajería y muchas herramientas que ayudarán a aumentar la productividad en el desarrollo de aplicaciones. El hecho de que J2EE se base en especificaciones públicas hace, tal y como se ha visto, que existan múltiples

implementaciones de servidores de aplicaciones. Actualmente, el mercado de los servidores de aplicaciones es uno de los mercados de productos software más activos y hay muchas empresas que lanzan sus productos y luchan por ofrecer cada vez más prestaciones. [3]

2.1.2. Patrones

2.1.2.1 ¿Que es un patrón?

Un patrón no es más que la descripción detallada de una solución adecuada a un problema concreto. Normalmente, un patrón está compuesto por el enunciado del problema y una o varias propuestas de solución. Para formalizar estas soluciones, se necesita una notación expresiva y rica en semántica que sea fácil de entender e interpretar.

2.1.2.2 Tipos de Patrón

Existen varios tipos de patrones, dependiendo del nivel de abstracción, del contexto particular en el cual aplican o de la etapa en el proceso de desarrollo. Estos son:

- ✓ *Patrones organizativos:* Describen la estructura y prácticas de las organizaciones humanas, especialmente las productoras de software.
- ✓ *Patrones de análisis:* Describen un conjunto de prácticas destinadas a elaborar el modelo y solución de un problema. Se usan para construir modelos conceptuales de software.
- ✓ *Patrones de arquitectura*: Describen la estructura general de un sistema. Identifica sus módulos, responsabilidades, y la colaboración entre dichos módulos.

---

[3] *Diseño e implementación de un framework de presentación para aplicaciones JEE http://openaccess.uoc.edu/webapps/o2/bitstream/10609/876/1/00765tfc.pdf*

- ✓ *Patrones de diseño:* Definen aspectos como interfaces, componentes, distribución, extensión.
- ✓ *Patrones de programación:* Pretenden describir como implementar ciertas tareas usando un lenguaje de programación concreto.
- ✓ *Patrones de seguridad*: describen mecanismos de seguridad.
- ✓ *Patrones de Integración de Aplicaciones*: Cada vez más es la regla, encontrarse con aplicaciones disímiles, de distintos fabricantes, en distintas plataformas, y que no se conectan entre sí. Aquí es donde este tipo de patrones pretenden facilitar las tareas de integración entre aplicaciones.

2.1.2.3 Patrones Arquitectónicos

Las aplicaciones Web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dando servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. Esto exige reflexiones sobre la mejor arquitectura y las técnicas de diseño más adecuadas. A continuación se realizará una breve descripción de las arquitecturas más comunes que se puede encontrar en este tipo de aplicaciones.

2.1.2.3.1 Arquitectura en 3 capas

En este tipo de arquitectura el reparto de responsabilidades es, en primera instancia, lógico. Y aunque posiblemente también sea físico, eso no es obligatorio. Las tres capas son:

- ✓ *Presentación*, recibe eventos del usuario a través de la interfaz presentada, y también formatea los resultados a desplegar.
- ✓ *Negocio*, el dominio del problema de negocio por el cual se tiene que desarrollar la aplicación
- ✓ *Acceso a Datos*, lógica que lleva a trae información entre la capa de negocio y los repositorios o sistemas externos donde los datos se almacenan (conectores, pools de conexiones, lógica de paginado, cachés...).

Figura II.4: Arquitectura en 3 capas

Aplicando este estilo arquitectónico se consigue mejorar la mantenibilidad y reusabilidad (soporte a cambios tecnológicos en una capa).

2.1.2.3.2 Arquitectura en 3 partes

Aquí sí existe una separación física en procesos, es decir, la ejecución está distribuida. Las tres partes son:

- ✓ *Cliente o Front-End,* que engloba la infraestructura de hardware y software donde se ejecuta la interfaz de usuario. También se suele referir a ellos como Canales.

- ✓ *Middleware*, capaz de recibir a través de la red, y mediante uno o varios protocolos de transporte (HTTP, TCP, etc.), uno o varios protocolos de mensajes (XML, SOAP, propietarios, etc.) requerimientos desde los distintos canales, habilitándose así el concepto de Arquitectura Multicanal.
- ✓ *Back-End*, normalmente una base de datos, aunque definitivamente puede ser otro proceso, incluso mucho más complejo que nuestra aplicación entera.

Normalmente las motivaciones que llevan a aplicar este tipo de arquitectura tienen que ver con Escalabilidad, Seguridad y Tolerancia a fallos.

2.1.2.3.3 Arquitectura en n capas y m partes

Una aplicación puede tener, a la vez, Arquitectura en n capas y en m partes, sean n y m igual a 3 o no. No se trata de uno o el otro. Es eventualmente uno indistintamente de la presencia o no del otro. Algunas de las capas que se puede encontrar son:

- ✓ *La capa del cliente* donde se consumen y presentan los modelos de datos. Para una aplicación Web, la capa cliente normalmente es un navegador Web. Los clientes pequeños basados en navegador no contienen lógica de presentación.
- ✓ *La capa de presentación* está contenida, parte en el navegador cliente (lógica en JavaScript...), parte en el middleware servidor (páginas Velocity, ASP.NET, JSP, JSF, etc.). Sabe cómo procesar una petición de cliente, cómo interactuar con la capa de lógica de negocio, y cómo seleccionar la siguiente vista a mostrar.
- ✓ *La capa de negocio* pueden ser clases o clases complejas como EJBs, Serviced Components. Esta capa, puede ejecutarse en el mismo middleware o, por cuestiones

de rendimiento, parte de la lógica de negocio se puede replicar en otras partes (tiers). Comúnmente, contiene los objetos y servicios de negocio de la aplicación. Recibe peticiones de la capa de presentación, procesa la lógica de negocio basada en las peticiones. Sus componentes se benefician de la mayoría de los servicios a nivel de sistema como el control de seguridad, de transacciones y de recursos.

- ✓ *La capa de integración* es el puente entre la capa de lógica de negocio. Encapsula la lógica para interactuar con la capa EIS. Algunas veces a la combinación de las capas de integración y de lógica de negocio se le conoce como capa central.
- ✓ *Los datos de la aplicación persisten en la capa EIS*. Contiene bases de datos relacionales, bases de datos orientadas a objetos, y sistemas antiguos.

Una arquitectura multicapa particiona todo el sistema en distintas unidades funcionales esto asegura una división clara de responsabilidades y hace que el sistema sea más mantenible y extensible. Los sistemas con multicapas se han probado como más escalables y flexibles que un sistema cliente /servidor, en el que no existe la capa central de lógica de negocios.

2.1.2.3.4 Arquitectura SOA (Service Oriented Architecture)

Su objetivo es concebir las aplicaciones desde otro punto de vista, una aplicación orientada a servicios combina datos en tiempo real con otros sistemas capaces de fusionar los procesos de negocio. Las aplicaciones basadas en SOA utilizan tecnología totalmente estándar como es XML y Servicios Web para la mensajería.

Realizando aplicaciones orientadas a servicio se pueden conectar aplicaciones heterogéneas con el aumento de flexibilidad que supone, y un punto muy importante es que permite que

las organizaciones interactúen cuando realmente lo requieran, sin necesidad de tener conexiones permanentes. Una aplicación SOA se la puede dividir en tres capas.

- ✓ *La capa de recepción de peticiones*: servicios controladores: encargados de recibir las peticiones de los clientes y realizar las llamadas necesarias a otros servicios, en la secuencia adecuada, para devolver una respuesta.
- ✓ *La capa de tareas*: servicios de negocio: representan una tarea de negocio.
- ✓ La capa de lógica reutilizables: servicios de utilidad: representan tareas muy reutilizables como tareas de negocio, acceso a datos.

Figura II.5 : Arquitectura SOA

2.1.2.3.5 Modelo – Vista – Controlador (MVC)

Se realiza una mención especial de este patrón por ser uno de los más famosos y extendido. Este patrón está asociado a 3 capas, se podría decir que suele estar presente allí donde estén 3 capas, pero es más bien una distribución fina de la secuencia de ejecución entre que se produce un evento en la capa de presentación y este es atendido en forma completa. Sus partes son.

- ✓ *Vista*, el componente que recibió el estímulo (un botón en un formulario, un check box, etc.) y generó un evento que puede involucrar el estado de los otros controles del formulario. Pertenece a la capa de presentación y está enteramente en el cliente y suele estar presente en parte del middleware.
- ✓ *Modelo*, componente asociado a entidades del dominio. No los procesos de negocio pero sí las entidades. Respecto de la arquitectura en 3 capas, entonces, el modelo incluye parte de la capa de negocio (entidades, no procesos) y toda la capa de acceso a datos. Respecto de la arquitectura en 3 partes, el Modelo se despliega (deployment) sobre el back-end entero, y también parte en el middleware
- ✓ *Controlador*, el componente que falta: el que se asocia a los procesos de negocio (el Modelo sólo se intercepta con las entidades de negocio). El Controlador recibe el evento que la vista generó y moviliza procesos del negocio que terminan cambiando el estado en el Modelo. Estos procesos de negocio suelen estar alineados a los casos de uso de la aplicación. Respecto de la arquitectura en 3 capas, el Controlador está comúnmente en la capa de negocio aunque puede tener una participación menor en la capa de presentación (según cada canal cliente, la pieza que recibe el evento que genera la Vista). Respecto de las 3 partes, el controlador suele estar normalmente en el middleware aunque en menor tiene cierta presencia en el front-end.

Este patrón puede ser el punto de partida de muchas aplicaciones Web, pero en muchas ocasiones no resulta suficiente para la aplicación que se necesita desarrollar. En muchas ocasiones hay que ir más allá, entrando en un diseño arquitectónico basado en n capas o m partes, con patrones de diseño diferentes para cada una de ellas.

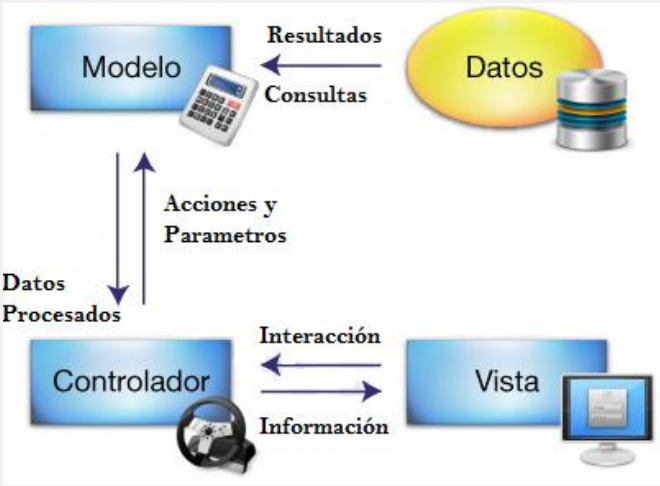


Figura II.6 : Modelo – Vista – Controlador (MVC)

También se puede encontrar la situación en que se necesita una arquitectura diferente para un tipo de aplicaciones específicas. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales. Pretender evitar las constantes recargas de página, ya que desde el principio se carga toda la aplicación y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos.

2.1.2.4 Patrones J2EE para la capa de presentación

- ✓ *Decorating o intercepting Filter*: Un objeto que está entre el cliente y los componentes Web. Este procesa las peticiones y respuestas.
- ✓ *Front Controller/ Front Component:* Un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. El patrón Front Controller podría dividir la funcionalidad en 2 diferentes objetos: el Front Controller y el Dispatcher. En

ese caso, El Front Controller acepta todos los requerimientos de un cliente y realiza la autenticación, y el Dispatcher direcciona los requerimientos a manejadores apropiada.

- ✓ *View Helper:* Un objeto helper que encapsula la lógica de acceso a datos en beneficio de los componentes de la presentación. Por ejemplo, los JavaBeans pueden ser usados como patrón View Helper para las páginas JSP.
- ✓ *Composite view:* Un objeto vista está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include o el action include es un patrón Composite View.
- ✓ *Service To Worker*: Es como el patrón de diseño MVC con el Controlador actuando como Front Controller pero,aquí el Dispatcher (el cual es parte del Front Controller) usa View Helpers a gran escala y ayuda en el manejo de la vista.
- ✓ *Dispatcher View*: Es como el patrón de diseño MVC con el controlador actuando como Front Controller pero con un asunto importante: aquí el Dispatcher (el cual es parte del Front Controller) no usa View Helpers y realiza muy poco trabajo en el manejo de la vista. El manejo de la vista es manejado por los mismos componentes de la Vista. [4]

2.1.3. Frameworks

Si se sigue una arquitectura basada en MVC, hay un conjunto de tareas repetitivas que se tienen que implementar para todas las aplicaciones.

Por ejemplo: recibir los parámetros de entrada de las peticiones y hacer validaciones, llamar a la lógica de negocio, escoger la siguiente vista para mostrar, etc.

---

[4] *Diseño de patrones J2EE*
http://java.ciberaula.com/articulo/diseno_patrones_j2ee/

Estas tareas se pueden implementar en un framework o marco de trabajo para la capa web que utilicen todos los desarrollos, de manera que no se deban implementar para cada aplicación.

Los desarrolladores crearán la capa web usando o extendiendo las clases e interfaces del framework. El uso de un framework para la capa web basado en una arquitectura MVC proporciona las ventajas siguientes:

- ✓ Desacopla la capa de presentación de la capa de negocio en componentes separados.
- ✓ Simplifica y estandariza la validación de los parámetros de entrada.
- ✓ Simplifica la gestión del flujo de navegación de la aplicación.
- ✓ Proporciona un punto central de control.
- ✓ Permite un nivel muy alto de reutilización.
- ✓ Impone la misma arquitectura para todos los desarrollos.
- ✓ Simplifica muchas tareas repetitivas.

2.1.3.1. ¿Qué es un framework Web?

Framework, se refiere a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se puede añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo.

Un framework Web, por tanto, se puede definirlo como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.

2.1.3.2. Origen de los frameworks para aplicaciones Web en Java

El surgimiento de los servlets de Java resultó en un avance bastante productivo en relación con el estándar CGI, ya que resultaba más poderosos y rápidos, así como portables y fácilmente entendibles; sin embargo, el desplegar código HTML en el explorador a través del método println(), especialmente cuando se trataba de gran cantidad de líneas, resultaba problemático y agotador.

Ante esta situación, llegaron al mundo los JSP (Java Server Pages), invirtiendo el concepto de los servlets, permitiendo insertar fácilmente código Java dentro de la Página HTML. Con esta solución, las aplicaciones Web adoptaron a los JSP como figura central, lo que pronto traería como consecuencia problemas en el control del flujo, así como el mantenimiento de páginas con demasiado código Java.

Ante esta nueva situación, se llegó a la idea de utilizar ambas tecnologías de manera conjunta, lo que represento una buena opción y satisfizo las necesidades de los desarrolladores aunque sólo por un tiempo ya que con la experimentación e implementación de dichas tecnologías, que además se presentaba como estándares, la comunidad que las había utilizado llegó a la conclusión de que lo que les ofrecía, o bien no

era suficiente para cumplir con los requerimientos de los diferentes proyectos, o lo realizaba de una manera no óptima o por debajo del nivel requerido.

Eso condujo a que los programadores desarrollan sus propios medios para cumplir con sus necesidades, lo que con el tiempo traería como resultado la aparición de los primeros frameworks orientados a las aplicaciones Web.

2.1.3.3. Características de los frameworks

A continuación se enuncia una serie de características que se puede encontrar en prácticamente todos los frameworks existentes.

- ✓ *Simplificación*: Al usar el modelo MVC, la estructura interna de las aplicaciones se simplifica. La consecuencia es la reducción de los errores, más facilidad de testeo, más fácil de mantener. Usando un framework se reduce considerablemente el tiempo de desarrollo de un proyecto, siempre que éste justifique su uso.
- ✓ *Unificador:* Se establece una estructura común para el desarrollo de aplicaciones, esto es básico cuando crece considerablemente el número de desarrolladores. Además se comparte la misma estructura de desarrollo en diferentes aplicaciones, esto hace que sea más fácil entender su funcionamiento evitando tener que el diseño arquitectónico de cada aplicación.
- ✓ *Integración*: Facilita la integración de aplicaciones que utilicen la misma arquitectura.
- ✓ *Abstracción de URLs y Sesiones*: No es necesario manipular directamente las URLs ni las sesiones, el framework ya se encarga de hacerlo.

- ✓ *Acceso a datos*: Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en BBDD, XML, etc.
- ✓ *Controladores*: La mayoría de frameworks implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.
- ✓ *Autentificación y control de acceso*: Incluyen mecanismos para la identificación de usuarios mediante login y clave pudiendo restringir el acceso a determinas páginas a determinados usuarios.[5]

2.1.3.4. Tipos de frameworks Web

Existen varios tipos de frameworks Web:

- ✓ Orientados a la interfaz de usuario, como Java Server Faces.
- ✓ Orientados a aplicaciones de publicación de documentos, como Coocon.
- ✓ Orientados a la parte de control de eventos, como Struts y algunos que incluyen varios elementos como Tapestry.

La mayoría de frameworks Web se encargan de ofrecer una capa de controladores de acuerdo con el patrón MVC, ofreciendo mecanismos para facilitar la integración con otras herramientas para la implementación de las capas de negocio y presentación.

---

[5] *Frameworks para aplicaciones Web en Java*
http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/viveros_s_ca/capitulo2.pdf

2.1.3.5. Ventajas y desventajas de un frameworks Web

2.1.3.5.1. Ventajas de un framework

Como ventajas en la utilización de un framework se tiene las siguientes:

a) Minimiza tiempos de desarrollo / acorta el "Time to market"

- ✓ Los proyectos de desarrollo ya no tendrán que resolver los múltiples problemas asociados a las aplicaciones web.
- ✓ Los frameworks reducen la codificación y sobretodo la puesta en marcha, ya que proporcionan subsistemas que sabemos que ya funcionan. En definitiva, proporcionan código que no se tendrá que mantener ni reescribir.

b) Reduce los riesgos del desarrollo

- ✓ Con un modelo de programación complejo como el de J2EE, el riesgo de fallos en los proyectos iníciales es alto. Un framework de desarrollo de aplicaciones reduce significativamente este riesgo al proporcionar una base fiable y suficientemente probada.

c) Proporciona una arquitectura consistente entre aplicaciones

- ✓ Al usar frameworks todas las aplicaciones generadas comparten una arquitectura común. Esto hace que sea más fácil de aprender, mantener y soportar.
- ✓ Cualquier programador que trabaje con un framework no deberá invertir gran parte de su tiempo en buscar las clases necesarias, interconectarlas o descubrir los métodos que contienen. Los frameworks ocultan toda esta complejidad dando un alto nivel de abstracción.

En definitiva, menor coste en el diseño, desarrollo, pruebas y mantenimiento de las aplicaciones.

2.1.3.5.2. Desventajas de un framework

Como desventajas en la utilización de un framework se tiene las siguientes:

a) *Limitación de la Flexibilidad*

Los componentes que un usuario construya a partir de un Framework deben residir dentro de las restricciones impuestas por la arquitectura del Framework

b) *Dificultad de Aprendizaje*

El usuario debe estudiar qué proporciona el Framework y cómo debe hacer uso de él. Este aprendizaje (requerido una sola vez) es costoso en el inicio, por partida, el desarrollo será más rápido.

c) *Reducción de la Creatividad*

Puede suceder que el usuario no pueda cambiar el comportamiento de una clase del Framework si ésta no permite que sus servicios puedan rescribirse o redefinirse. [6]

2.2. Frameworks de presentación Web en Java

En la actualidad existe un abanico muy amplio de opciones de frameworks Web para la capa de presentación en el entorno Java.

---

[6] *Modelos y métodos http://www.google.com.ec/url?sa=t&rct=j&q=Ventajas+y+desventajas+de+un+framework+Minimizar+tiempos+de+desarrollo+/+acorta+el+%E2%80%9CTime+to+market%E2%80%9D&source=web&cd=2&ved=0CCgQFjAB&url=http%3A%2F%2Fsubversion.assembla.com%2Fsvn%2Fpfc201011%2FMemoria%2520Proyecto%2F2%2520BloqueI%2FModelosYMetodos.docx&ei=t3E8T-L7B4rugget16iZCw&usg=AFQjCNHgWN6m3Wrp13Mg3V53_aH-n7dQKg*

Por lo que se ha elegido para el estudio preliminar los frameworks más populares según una encuesta en la Web, como son Struts2, JSF, Tapestry, Spring MVC y Cocoon.

Figura II.7 : Frameworks Web para J2EE

2.2.1. Struts2

Struts2 es un framework web de la familia de software libre, implementa el patrón de diseño MVC que nació de la necesidad de evolucionar el código de Struts con el objetivo de simplificar todavía más el desarrollo de la capa de presentación de las aplicaciones web. Durante el diseño de la siguiente versión de Struts, que iba a denominarse Struts Ti, se observaron objetivos y puntos en común con el framework WebWork2, por lo que las comunidades decidieron unir esfuerzos y fusionar estos dos proyectos dando lugar al proyecto Struts2.

2.2.1.1. Características

Las características principales de Struts2 y mejoras respecto a la versión anterior son:

- ✓ Framework MVC orientado a acciones.

- ✓ Permite utilizar cualquier clase Java normal (POJO) como acción.
- ✓ Arquitectura flexible de bajo acoplamiento mediante plugins e interceptores que permiten personalizar el tratamiento de las peticiones hasta llegar a cada acción de forma individual o para un conjunto de acciones.
- ✓ Conversión de tipos automática que mapea de forma transparente los valores HTTP a las acciones mediante setters, solucionando y simplificando uno de los mayores esfuerzos que se necesitan al crear aplicaciones web.
- ✓ Define un lenguaje OGNL compatible con JSTL que expone las propiedades de múltiples objetos como si fueran un único JavaBean.
- ✓ Ficheros de configuración modulares que permiten descomponer la configuración en varios archivos para mejorar la gestión de los mismos en proyectos grandes.
- ✓ Gran cantidad de etiquetas, desde simples campos de texto hasta calendarios para seleccionar fechas y vistas en árbol.
- ✓ Permite plugins para definir nuevos tipos de resultado de forma que se soportan múltiples tecnologías de presentación como JSP, FreeMarker, PDF, JasperReports, etc.
- ✓ Proporciona una colección de plugins opcionales que proporcionan funcionalidad como por ejemplo el upload de ficheros, prevenir el problema de repetición de posts del mismo formulario o gestionar la seguridad.

2.2.1.2. Ventajas

- ✓ Posee una arquitectura modular de acoplamiento que lo hace muy extensible.

- ✓ El número de acciones personalizadas para la generación de la vista y las capacidades de éstas han aumentado respecto a versiones anteriores del framework.
- ✓ La posibilidad de disponer de una serie de opciones de configuración predefinidas simplifica enormemente la labor del desarrollador.
- ✓ Este framework ofrece una mayor productividad en la construcción, desarrollo y mantenimiento de aplicaciones.
- ✓ Permite utilizar menos configuración XML mediante anotaciones, define comportamientos por defecto y se rige por convenios inteligentes.

2.2.1.3. Inconvenientes

- ✓ El hecho de no abarcar todas las capas de la aplicación web hace que el interfaz y estas capas no esté tan automatizado, convirtiendo los accesos a los datos (DAO) en monótonos de desarrollar.
- ✓ La curva de aprendizaje es lenta. [7]

2.2.2. Spring MVC

Spring Framework está diseñado como una serie de módulos que pueden trabajar independientemente uno de otro, lo que quiere decir que puedes utilizar únicamente los módulos que necesites. Además intenta mantener un mínimo acoplamiento entre la aplicación y el propio framework de forma que podría ser desvinculada de él sin demasiada dificultad.

---

[7] *Diseño e implementación de un framework de presentación para J2EE. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/6981/1/jcirianoTFC_memoria.pdf*

El sub-framework SpringMVC es solo una parte del diseño global, que va mucho más allá. Entre los principales módulos del framework Spring se tiene:

- ✓ **Core** - Es el núcleo en donde se encuentra el **BeanFactory** – el contenedor fundamental de Spring y quien se encarga de la inyección de dependencias.
- ✓ **DAO** - Provee una capa de abstracción sobre JDBC, abstrae el código de acceso a datos de una manera simple y limpia. Tiene una capa de excepciones sobre los mensajes de error provistos por cada servidor específico de base de datos. Además cuenta con manejo de transacciones a través de AOP.
- ✓ **ORM** – Provee la integración para las distintas APIs de mapeo objeto-relacional incluyendo JPA, JDO, Hibernate e iBatis.
- ✓ **AOP** – Provee la implementación de programación orientada a aspectos, permitiéndonos desarrollar interceptores de método y puntos de corte para desacoplar el código de las funcionalidades transversales.
- ✓ **Web** – Módulo que aporta clases especiales orientadas al desarrollo web e integración con tecnologías como Struts y JSF.
- ✓ **MVC** – una implementación del conocido patrón de diseño aplicando los principios de Spring MVC para construir aplicaciones Web.
- ✓ **JEE** – Provee integración con aplicaciones Java Enterprise Edition así como servicios JMX, JMS, EJB, etc. [8]

[8] *Introducción a Spring Framework Java*
http://picandocodigo.net/2010/introduccion-a-spring-framework-java/

2.2.2.1. Características

- ✓ La inicial motivación era facilitar el desarrollo de aplicaciones J2EE, promoviendo buenas prácticas de diseño y programación.
- ✓ Maneja la asignación de peticiones a controladores y desde estos a las vistas. Implica el manejo y validación de formularios.
- ✓ Actúa como pegamento de integración entre diferentes APIs (JDBC, JNDI, etc.) y frameworks (por ejemplo entre Struts e iBatis).
- ✓ Permite usar archivos XML de configuración, programación mediante la API y mediante un estándar JSR.
- ✓ Todos los componentes pueden ser testeados fuera del contenedor.

2.2.2.2. Ventajas

- ✓ Spring recomienda el uso de Interfaces, basta con reimplementar las interfaces.
- ✓ Se puede ejecutar dentro de un contenedor Web o fuera de él en un Swing.
- ✓ Ofrece una división limpia entre el Modelo-Vista-Controlador.
- ✓ Es muy flexible, ya que implementa toda su estructura mediante interfaces.
- ✓ Provee interceptores y controllers que permiten interpretar y adaptar el comportamiento común en el manejo de múltiples requests.
- ✓ Los controllers de Spring MVC se configuran mediante IoC como los demás objetos, lo cual los hace fácilmente testeables e integrables con otros objetos que estén en el contexto de Spring, y por tanto sean manejables por éste.
- ✓ Tiene una interfaz bien definida para la capa de negocio.

- ✓ Ofrece mejor integración con tecnologías distintas a JSP, como Velocity, XSLT, FreeMaker y XL.

2.2.2.3. Inconvenientes

- ✓ No se puede evaluar si un objeto ha sido bien inyectado más que en tiempo de ejecución.
- ✓ Mucha configuración XML
- ✓ El contenedor de Spring no es ligero [9]

2.2.3. JSF

JSF es una plataforma desarrollada a través del Java community Process por Sun Microsystems. JSF se establece como un estándar para la construcción de interfaces de usuario ubicándose en el lado del servidor. Con la contribución de un grupo de expertos, la API fue diseñada de forma tal de hacerla extensible y mejorada a través de herramientas que hagan aún más fácil el desarrollo de aplicaciones Web.

JSF se encarga de enriquecer los conceptos tradicionales relacionados con la interfaz de usuario Web debido a que maneja los componentes visuales como componentes encapsulados en clases que incluyen las funcionalidades internas de los mismos. De esta forma los componentes pueden ser visualizados en distintos dispositivos.

---

[9] *Introducción a Spring MVC*
http://jaimecarmonaloeches.blogspot.com/2012/01/introduccion-spring-mvc.html

Teniendo a la facilidad de uso como su principal objetivo la arquitectura provista por JSF define una separación de la lógica de la aplicación y la presentación pero sin dejar de lado la facilidad para la comunicación entre la capa de presentación y el código de la aplicación.

2.2.3.1. Características

- ✓ Framework MVC basado en modelo de componentes de interfaz de usuario.
- ✓ JSF incluye controles GUI, APIs y custom tags con los que se pueden crear interfaces personalizables y complejas.
- ✓ El framework proporciona eventos. El código puede gestionar y responder a diversos eventos como si fuera una aplicación de escritorio.
- ✓ Gestión de beans que permiten simplificar el tratamiento y almacenamiento de los parámetros de las peticiones.
- ✓ Validación de formularios y conversión de tipos automáticas.
- ✓ Proporcionan etiquetas que implementan funcionalidad mediante AJAX sin tener que manejar JavaScript y que permiten llamar a la lógica de negocio.
- ✓ Configuración centralizada y declarativa mediante archivos.
- ✓ Al ser un estándar se dispone de una gran variedad de librerías de componentes suministradas por diferentes proveedores.

2.2.3.2. Ventajas

- ✓ Proporciona una rica arquitectura para manejar el estado de los componentes, procesar datos, validación de usuarios y manejar eventos.

- ✓ Permite separar claramente el contenido de la presentación y de la lógica.
- ✓ Es una especificación, lo que permite tener varias implementaciones (tanto de código cerrado como de código abierto).
- ✓ Permite modificar el comportamiento de la aplicación sin conocer el lenguaje en el que está implementado.
- ✓ No es necesario conocer el framework en detalle para poder comenzar a utilizarlo.
- ✓ Comunidad y herramientas de soporte en aumento.

2.2.3.3. Inconvenientes

- ✓ La creación de componentes propios es compleja.

2.2.4. Tapestry

Tapestry es un framework de código abierto para la creación de aplicaciones web de forma dinámica, robusta y altamente escalable en Java. Tapestry complementa y construye desde el estándar Java Serlvet API, funcionando también en cualquier servidor contenedor de servlets o contenedor de aplicaciones. Desarrollar aplicaciones Tapestry implica crear plantillas HTML usando HTML plano, y combinando las plantillas con pequeños trozos de código Java usando un descriptor de archivos XML (opcional).

Tapestry divide una aplicación web en un conjunto de páginas, cada una compuesta de componentes. Esto le otorga una estructura consistente, permitiendo a Tapestry asumir responsabilidades clave como la construcción y envió de URLs, almacenamiento del estado

persistente en el cliente o en el servidor, validación de entradas de usuario, localización/internacionalización, y reporte de excepciones. En Tapestry, tu creas tus aplicaciones en términos de objetos, y los métodos y propiedades de estos objetos, y no especificando términos de URLs y términos de consulta. Tapestry ofrece un desarrollo realmente orientado a objetos a las aplicaciones web Java.

2.2.4.1. Características

- ✓ Sigue el enfoque POJO (Plain Old Java Object).
- ✓ Además de los componentes proporcionados por el framework permite que el desarrollador construya fácilmente sus propios componentes a medida
- ✓ Contiene componentes para interacciones AJAX
- ✓ Soporte para internacionalización
- ✓ Soporte para validación de datos de entrada
- ✓ Soporte para Inyección de Dependencias
- ✓ Integración con Hibernate, Spring.
- ✓ Útil cuando la lógica de negocio es más compleja.
- ✓ Soporte para implementar aplicaciones Web CRUD directamente con Hibernate

2.2.4.2. Ventajas

- ✓ Simplicidad en la creación de aplicaciones web. Facilita enormemente el desarrollo de interfaces ricas e interactivas. Construir componentes con Tapestry es tan fácil como construir una página. La idea es muy simple: una platilla HTML para definir

la apariencia; una especificación para indicar qué componentes se usan en la página y cómo se relacionan sus parámetros y una implementación para programar métodos oyentes y proporcionar los datos necesarios.

- ✓ Facilita la reusabilidad y el mantenimiento en las aplicaciones web: una vez diseñados los componentes necesarios, construir una aplicación no es más que unirlos. Tapestry es un framework de presentación realmente basado en componentes ya que es de los frameworks que mejor representa el concepto de componente, llevándolo hasta el límite de que en él todo son componentes.
- ✓ Consistencia a la hora de que distintos desarrolladores pueden encontrar soluciones similares a problemas similares.
- ✓ Eficiencia para que las aplicaciones sean fácilmente escalables.
- ✓ Reacción ante los errores, aportando modos de diagnósticos.
- ✓ Tapestry ya forma parte del proyecto Apache, lo cual es una garantía.

2.2.4.3. Inconvenientes

- ✓ No existe bibliografía sobre él. Actualmente no hay en el mercado ningún libro sobre Tapestry, aunque Howard M. Lewis Ship, padre de Tapestry, ha terminado de escribir el primero: Tapestry in Action, que será publicado por Manning Publications el próximo otoño.
- ✓ La documentación que acompaña al framework está algo desactualizada. Con la entrada de Tapestry en Apache y la redacción del libro sobre Tapestry muchos de los cambios que han tenido lugar en el framework no han sido documentados en los

manuales, aunque su contenido sigue siendo de gran utilidad y ha sido una gran fuente de información para la realización de este proyecto.

- ✓ Muchos desarrolladores no ven con buenos ojos que Tapestry no use JSP. En los equipos de desarrollo existe una formación importante en JSP y no ven conveniente desechar ese conocimiento para aprender una nueva tecnología.
- ✓ Los desarrolladores están acostumbrados a pensar en términos de URL's, parámetros codificados dentro de ellas, HttpSession, y demás complicaciones de la API Servlet, por lo que pasar a pensar en términos de componentes puede resultar duro.
- ✓ La curva de aprendizaje de Tapestry es bastante pronunciada. Esto es debido principalmente a lo comentado en el punto anterior

2.2.5. Cocoon

Cocoon es un framework de desarrollo Web de publicación Web, basado en componentes y en la Separación de Intereses. Cuenta con desarrollo total en Java por lo cual se puede ejecutar desde cualquier servidor que pueda contener Servlets; y al ser un Servlet cuenta con las ventajas de éstos, es decir, se ejecutan como threads de forma simultánea en el mismo contexto y no tienen que llamar a métodos auxiliares como lo hacen tecnologías del estilo CGI.

2.2.5.1. Características

- ✓ Provee un control de flujo avanzado permitiendo describir el orden en que las páginas deben ser enviadas al cliente.

- ✓ Es bastante configurable y personalizable.
- ✓ Adopta características para escribir páginas de servidor en XML (XSPs).
- ✓ Permite diferenciar el procesamiento del documento para tenerlo en distintos formatos, dependiendo del tipo de software que hace la petición y cuenta con un sistema de caché para tener un mejor rendimiento.
- ✓ Separación de contenido, presentación y lógica en su aplicación.

2.2.5.2. Ventajas

- ✓ Permite separar claramente el contenido de la presentación y de la lógica.
- ✓ Permite modificar el comportamiento de la aplicación sin conocer el lenguaje en el que está implementado.

2.2.5.3. Inconvenientes

- ✓ Requiere conocimiento avanzados de XML, hojas de estilo XSL.
- ✓ Requiere amplios conocimientos de hojas de estilo XSL.
- ✓ Comunidad relativamente pequeña.
- ✓ Curva de aprendizaje elevada.
- ✓ La transformación de XMLs requiere bastante capacidad de proceso
- ✓ Existe poca documentación. [10]

[10] *Apache Tapestry*
http://wikis.uca.es/wikiii/index.php/Apache_Tapestry

CAPÍTULO III

ANÁLISIS COMPARATIVO DE FRAMEWORKS WEB EN JAVA

3.1. Introducción

Hoy en día la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE, mientras más pasa el tiempo mejora en cada una de sus características.

Tiene un gran valor echar un vistazo a los frameworks más populares que existen actualmente, que sean capaces de llevar a cabo la consecución de los objetivos propuestos para la elaboración de este proyecto.

El presente capitulo contiene el análisis comparativo de los frameworks Web para Java más populares que se puede encontrar en el mercado, que tiene la finalidad de tomar una decisión basada en parámetros de evaluación, permitiendo de esta manera determinar cuál es la más apropiada para la implementación Web.

3.2. Elección de frameworks a evaluar.

Para seleccionar los frameworks Web para su evaluación se basa en el Sitio Web "Java México| user group" de acuerdo a una encuesta con la interrogante ¿Qué framework MVC prefieres para desarrollar?.

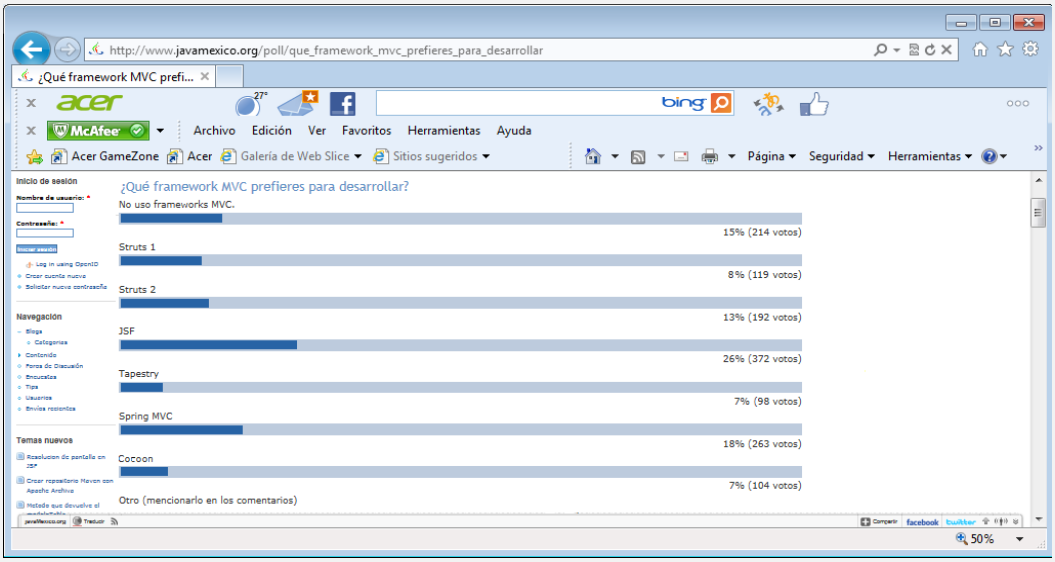


Figura III.8: Página del Sitio Web con encuesta de los frameworks MVC

A esta interrogante respondieron un total de 1445 participantes en la que proyecta los siguientes datos:

TABLA III.I

Tabulación de encuesta de framewoks MVC

Framework MVC	Votos	Porcentaje
No usa Frameworks	214	15
Struts 1	119	8
Struts 2	192	13

JSF	372	26
Tapestry	98	7
Spring MVC	263	18
Cocoon	104	7
Otros	83	6
TOTAL	**1445**	**100**

Fuente: http://www.javamexico.org/poll/que_framework_mvc_prefieres_para_desarrollar
Elaborado por: Ximena Moposita, Mercedes Morán

En esta encuesta se mencionan algunos frameworks MVC, los mismos que se han tomado en cuenta para la investigación, ya que se consideran los más usados por los programadores orientados a la Web, se ha llegado a esta interpretación ya que en la opción Otros se muestra una mínima cantidad de votos. Los frameworks MVC elegidos para el estudio preliminar son: Struts2, JSF, Tapestry, Cocoon y Spring MVC, para luego seleccionar dos de ellos y analizarlos más detenidamente.

3.2.1. Enfoque de análisis de los cinco frameworks seleccionados

3.2.1.1. Determinación de los parámetros de evaluación de los frameworks

Se definen a continuación los parámetros que permiten evaluar los frameworks que permiten el desarrollo de aplicaciones Web en Java que es el objeto de estudio de este trabajo investigativo.

3.2.1.1.1. Selección de los parámetros para el análisis de los frameworks

TABLA III.II

Parámetros a evaluar

Parámetros	Variable	Indicadores
1. Producto	Madurez del producto	✓ Tiempo en el mercado. ✓ Versiones del producto. ✓ Fuentes de empleo
2. Facilidades	Facilidades para el desarrollo	✓ Soporte de plugins y con otras tecnologías. ✓ Documentación. ✓ Comunidad. ✓ Configuración.

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.2. Valoración de parámetros.

Para analizar y evaluar los frameworks Struts2, Spring MVC, JSF, Tapestry y Cocoon se ha tomado los siguientes valores con los que se calificara cada uno de los parámetros.

TABLA III.III

Valoración de parámetros

Valor	Significado
1	Malo
2	Regular
3	Bueno

4	Excelente

Elaborado por: Ximena Moposita, Mercedes Morán

Para obtener resultados cualitativos y cuantitativos y poder llegar a una conclusión final de los frameworks evaluados se utilizará en el proceso de análisis comparativo la siguiente fórmula:

$$CA = \sum_{i=1}^{n} VPEi$$

Donde:

VPE: Representa al valor del parámetro evaluado.

i: Representa al parámetro evaluado

n: Total de parámetros evaluados

CA: Calificación total del parámetro

3.2.1.3. Comparación de los frameworks a evaluar

3.2.1.3.1. Producto

3.2.1.3.1.1. Análisis de los indicadores de Madurez del Producto

3.2.1.3.1.1.1. Tiempo en el mercado y Versiones del producto

Analiza si el framewok es suficientemente sólido, efectivo estableciendo el tiempo que se encuentra en el mercado tomando experiencia y estableciendo las versiones que fueron lanzadas al mercado con sus nuevas mejoras.

TABLA III.IV

Comparación de los indicadores Tiempo en el mercado y Versiones del producto

FRAMEWORK	TIEMPO EN EL MERCADO Y VERSIONES DEL PRODUCTO
Struts 2	Este framework se encuentra en el mercado desde el 2001 siedo usado por grandes y pequeñas empresas en una gran cantidad de proyectos. Teniendo las siguientes versiones: ✓ Struts 2.0.0 hasta Struts 2.0.9; Struts 2.1.2 hasta Struts 2.1.8; Struts 2.2.x; Struts 2.3.x; La última versión Struts 2.2.3.2 lanzada el 23 de enero de 2012.
Spring MVC	Su lanzamiento se dio a partir del 2004 siendo muy flexible capaz de adaptar los requerimientos a grandes y pequeños proyectos. Spring MVC posee las siguientes versiones: ✓ Spring MVC 2.5; Spring MVC 3.0; Última versión Spring MVC 3.1.0 puesta en marcha el 13 de diciembre de 2011.
JSF	JSF se encuentra en el mercado desde el 2004, desde ese entonces no ha parado de crecer y poco a poco se ha vuelto popular entre los programadores Web. Teniendo las siguientes versiones en el mercado: ✓ JSF 1.0; JSF 1.1; JSF 1.2; JSF 2.0; JSF 2.1; Su última versión JSF 2.1.7 a partir de 10 de febrero 2012.
Tapestry	Este framework fue lanzado al mercado en el año 2000, su demanda ha sido escasa ya que de versión a versión cambia drásticamente. Posee

	las siguientes versiones: ✓ Tapestry 2.x; Tapestry 3.x; Tapestry 4.x; Tapestry 5.x; Su última versión 5.3.2 lanzada el 7 de febrero de 2012.
Cocoon	Este framework a parece en el mercado en 1999, es muy escasa su demanda y no es popular entre los programadores Web. Se encuentra disponibles las siguientes versiones: ✓ Cocoon 1.x; Cocoon 2.0 hasta Cocoon 2.2; La versión más actualizada Cocoon 3 beta publicada el 09 de junio 2011.

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.3.1.1.2. Fuentes de empleo

Con este indicador se analiza la demanda que existe por parte de las empresas en el Ecuador al solicitar profesionales con conocimientos en estos frameworks.

TABLA III.V

Comparación del indicador Fuentes de empleo

Framework MVC	Total de Empleos en el mes de febrero de 2012
Struts 2	18
Spring MVC	20
JSF	21
Tapestry	2
Cocoon	0
TOTAL	**61**

Fuente: http://es.jobrapido.com/?w=jsf&l=españa&r=auto

3.2.1.3.1.2. Resultados

TABLA III.VI

Resultados de la variable Madurez de Producto

Frameworks / **Indicadores**	Struts 2	Spring MVC	JSF	Tapestry	Cocoon
Tiempo en el mercado	3	3	4	2	1
Versiones del producto	4	4	4	2	3
Fuentes de empleo	3	4	4	1	1
Total	**10**	**11**	**12**	**5**	**5**

Elaborado por: Ximena Moposita, Mercedes Morán

$$\mathbf{C_{Struts2}} = \sum_{i=1}^{3} \text{VPEi} = 3 + 4 + 3 = 10$$

$$\mathbf{C_{SpringMVC}} = \sum_{i=1}^{3} \text{VPEi} = 3 + 4 + 4 = 11$$

$$\mathbf{C_{JSF}} = \sum_{i=1}^{3} \text{VPEi} = 4 + 4 + 4 = 12$$

$$\mathbf{C_{Tapestry}} = \sum_{i=1}^{3} \text{VPEi} = 2 + 2 + 1 = 5$$

$$\mathbf{C_{Cocoon}} = \sum_{i=1}^{3} \text{VPEi} = 1 + 3 + 1 = 5$$

3.2.1.3.1.3. Interpretación

Strust 2 es un framework que se encuentra en el mercado 11 años considerándose solido y efectivo al momento de crear grandes y pequeños proyectos web.

Spring MVC tiene una arquitectura sólida y sobre todo muy flexible capaz de adaptar cualquier tipo de requerimiento a proyectos web. A estado disponible durante 8 años igualmente que el framework JSF que se introdujo rápidamente en el mercado ya que tiene una solución correcta y robusta y por esta razón muchas empresas lo utilizan actualmente.

Tapestry y Cocoon que tienen más antigüedad entre estos 5 frameworks no han alcanzado la madurez necesaria. Tapestry posee demasiadas versiones que cambian sin respetar la compatibilidad de la versión anterior. Luego de 12 años ya debería tener una madurez adecuada pero sigue cambiando. Cocoon en algunos temas todavía va evolucionando, no siendo estable en todas sus áreas de uso.

3.2.1.3.2. Facilidades

3.2.1.3.2.1. Análisis de los indicadores de facilidades para el desarrollo

3.2.1.3.2.1.1. Soporte de plugins y otras tecnologías

Indicador que analiza el criterio que busca la posibilidad de mejorar la aplicación añadiendo funcionalidades adicionales distintas a los nativos de cada framework.

TABLA III.VII

Comparación del indicador Soporte de plugins y con otras tecnologías

FRAMEWORK	SOPORTE DE PLUGINS Y OTRAS TECNOLOGÍAS
Struts 2	El comportamiento de Struts2 puede ser mejorado y aumentado por el uso de plugins que permiten personalizar el tratamiento de las

	peticiones hasta llegar a cada acción o conjunto de acciones. Los plugins de Struts 2 no tienen que ser declarados ni configurados de ninguna forma. Basta con agregar al **classpath** del jar que lo contiene. Struts 2 permite la integración con otras tecnologías que permiten realizar testeabilidad, sistemas de plantillas y ORM.
Spring MVC	Es muy extensible y acoplable con múltiples tecnologías y se puede realizar testeabilidad, sistemas de plantillas y ORM, pero no posee soporte para plugins ya que es un framework altamente configurable vía interfaces.
JSF	JSF cuenta con una arquitectura y modelo extensible muy sencillo y bien definido que ha estado presente desde el principio que permite a las partes principales del framework ampliar o sustituir cuando sea necesario con la utilización de plugins.
Tapestry	Cuenta con soporte a plugins permitiéndole extender nuevas funcionalidades para aumentar mejoras a la aplicación.
Cocoon	Ofrece una arquitectura extensible permitiendo de esta manera la posibilidad de mejorar la aplicación añadiendo nuevas funcionalidades con plugins; puede que tenga que crear un entorno.

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.3.2.1.2. Documentación

Permite valorizar la cantidad y calidad de tutoriales existentes para el entendimiento profundo de cada framework a evaluar.

TABLA III.VIII

Comparación del indicador documentación

FRAMEWORK	DOCUMENTACIÓN
Struts 2	Existen tutoriales que permiten ayudar al desarrollo de páginas web basadas en este framework. *Google:* Tutoriales de Strust2 con 1020,000 resultados.
Spring MVC	Tiene una buena y gran documentación que cubre todos sus ámbitos. *Google*: Tutoriales de Spring MVC con 618,00 resultados.
JSF	Posee una excelente documentación para la ayuda a sus seguidores para un buen entendimiento, al momento de desarrollar páginas web. *Google*: Tutoriales de JSF con 1750,000 resultados.
Tapestry	La documentación no es abundante y actualizada, pero la poca documentación que existe se encuentra detallada paso a paso. *Google*: Tutoriales de Tapestry con 434,000 resultados.
Cocoon	Cuenta con documentación escasa, desactualizada e incompleta, no profundiza en cosas puntales, lo que puede ser un problema para alguien que esté empezando. *Google*: Tutoriales de Cocoon con 200,000 resultados.

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.3.2.1.3. Comunidad

Se valoriza la existencia de foros, wikis, Blogs, artículos de cada uno de los frameworks.

TABLA III.IX

Comparación del indicador Comunidad

FRAMEWORK	COMUNIDAD
Struts 2	La comunidad de este framework es grande debido a su difusión, que aportan con artículos, foros, wikis permitiendo de esta manera dar una solución a cualquier problema que pueda surgir durante el desarrollo. *Google*: Comunidad de Strust 2 con 20,600resultados
Spring MVC	Tiene una nutrida comunidad de usuarios que aportan artículos y trabajos ayudando de esta manera a resolver problemas. *Google*: Comunidad de Spring MVC con 9,660 resultados
JSF	Cada día la comunidad de este framework va en aumento aportando con novedades que son útiles al momento de desarrollar con JSF. *Google*: Comunidad de JSF con 230,000 resultados
Tapestry	Dispone de una comunidad mediana, no cuenta con suficientes recursos de ayuda a las personas que utilizan este framework. *Google*: Comunidad de Tapestry con 8470 resultados
Cocoon	Posee una comunidad relativamente pequeña; no existen artículos, foros, etc. que permitan resolver algunos inconvenientes. *Google*: Comunidad de Cocoon con 6770 resultados

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.3.2.1.4. Configuración

Parámetro que evalúa la sencillez de cómo se configura (XML, anotaciones) el framework para que esté listo al momento de utilizarlo.

TABLA III.X

Comparación del indicador configuración

FRAMEWORK	CONFIGURACION
Struts 2	La configuración de este framework se lo realiza en el fichero central de configuración strust.xml en el mismo que se puede agregar archivos XML. Este fichero se organiza en paquetes donde cada paquete tiene las acciones permitiendo que de esta forma el fichero no crezca y sea inmanejable. Struts 2 soporta el uso de anotaciones y proporciona una forma de hacer la configuración de manera automática si se hace uso de una serie de convenciones.
Spring MVC	Potente y sencillo de configuración tanto el framework y las clases de la aplicación como JavaBeans. Su configuración está basada en distintos ficheros XML. SpringMVC se puede configurar adicionalmente con el uso de anotaciones evitando el trabajo tedioso o repetitivo el cual hace que la implementación de las mismas sea más sencilla, rápida y con menos complicaciones.
JSF	La configuración de JSF contendrá el archivo faces-config.xml en el mismo que se agregan los beans y las reglas de navegación. Una de las características de JSF son anotaciones en lugar del uso de

	un fichero XML haciéndole de esta forma que la configuración de aplicaciones sea rápida y sencilla.
Tapestry	Tapestry se configuran casi en su totalidad en Java, Sin embargo, una cantidad pequeña pero necesaria de la configuración se produce en el interior del descriptor del servlet de Tapestry en su fichero web.xml, el mismo que se lo hace uno para cada componente.
Cocoon	Cocoon cuenta para su configuración con el fichero sitemap.xml que contiene los componentes utilizados por las pipelines y las pipelines para el funcionamiento de la aplicación.

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.3.2.2. Resultados

TABLA III.XI

Resultados de la variable de Facilidades de desarrollo

Frameworks / **Indicadores**	Struts 2	Spring MVC	JSF	Tapestry	Cocoon
Soporte de plugins y otras tecnologías	4	3	3	4	4
Documentación	4	4	4	2	1
Comunidad	3	3	4	2	1
Configuración	4	4	4	3	3
Total	**15**	**14**	**15**	**11**	**9**

Elaborado por: Ximena Moposita, Mercedes Morán

$$C_{Struts} = \sum_{i=1}^{4} VPEi = 4 + 4 + 3 + 4 = 15$$

$$C_{SpringMVC} = \sum_{i=1}^{4} VPEi = 3 + 4 + 3 + 4 = 14$$

$$C_{JSF} = \sum_{i=1}^{4} VPEi = 3 + 4 + 4 + 4 = 15$$

$$C_{Tapestry} = \sum_{i=1}^{4} VPEi = 4 + 2 + 2 + 3 = 11$$

$$C_{Coocoon} = \sum_{i=1}^{4} VPEi = 4 + 1 + 1 + 3 = 9$$

3.2.1.3.2.3. Interpretación

Al momento de crear aplicaciones Web una de las tareas más importantes son las facilidades de desarrollo en donde Strust 2 cuenta con una documentación y comunidad necesaria para obtener conocimientos y tener la capacidad de desarrollar una aplicación Web, posee la habilidad de extender funcionalidades nuevas y cuenta con una configuración aceptable.

Spring MVC no consta con un soporte de plugins en cambio se integra fácilmente a otras tecnologías mediante configuraciones, en cuanto a la documentación y comunidad es aceptable permitiendo tener el conocimiento necesario de este framework.

JSF cuenta con la suficiente información y comunidad a través del internet que pueden dar soporte a las diferentes consultas que tengan los desarrolladores web, se puede integrar a otras tecnologías permitiendo mejorar el desarrollo de las aplicaciones web.

Tapestry y Cocoon no cuentan con la información necesaria en el internet ni presenta una comunidad amplia de desarrolladores volviéndose esto un problema, estos frameworks admiten pocas tecnologías añadir para mejorar su funcionalidad y permiten una configuración relativamente sencilla.

3.2.1.4. Resultados del análisis realizado

TABLA III.XII

Resultado Final

Frameworks / **Parámetros**	Struts 2	Spring MVC	JSF	Tapestry	Cocoon
Producto	7	7	8	4	4
Facilidades	15	14	15	11	9
Total	**22**	**21**	**23**	**15**	**13**

Elaborado por: Ximena Moposita, Mercedes Morán

3.2.1.4.1. Diagrama general con porcentajes de resultados

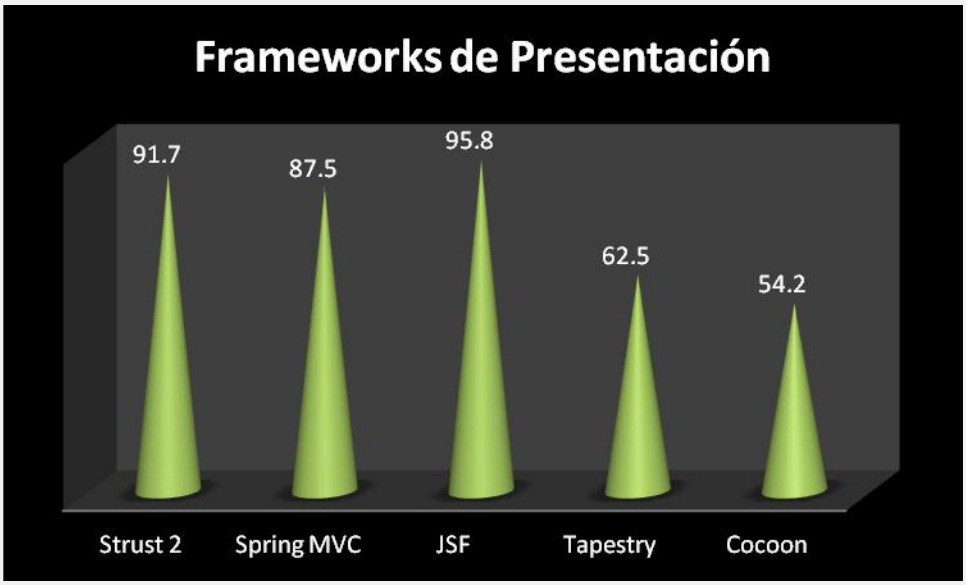


Figura III.9 : Porcentajes de resultados obtenidos

3.2.1.4.2. Conclusiones de los resultados obtenidos.

Una vez realizado este análisis preliminar se puede decir que de los 5 frameworks estudiados los que presentan mejores características desde este punto de vista al momento de desarrollar una aplicación Web son: JSF, Struts 2, y Spring MVC.

Se selecciona los frameworks de presentación Struts 2 y Spring MVC para hacer un estudio más profundo debido a que ya existen tesis que abarcan una amplia investigación de JSF.

3.3. Estudio de Frameworks a comparar

3.3.1. Struts 2.

Figura III.10 : Logotipo de Struts 2

3.3.1.1. Introducción

Las cada vez más exigentes demandas de las aplicaciones Web modernas han llevado al equipo de desarrollo de Struts, en su afán de mejorar continuamente las prestaciones del framework, a replantear la estructura y composición del mismo, adoptando un nuevo enfoque que resulte más flexible y sencillo de utilizar. En esta línea nace Struts 2.

Más que una nueva versión de Struts, Struts 2 es realmente la fusión de dos frameworks: por un lado el clásico Struts con nuevas capacidades añadidas, y por otro WebWork, un framework desarrollado por OpenSymphony que proporciona un fuerte desacoplamiento entre las capas de la aplicación.

El resultado es un producto que integra las capacidades de ambos frameworks, lo que sin duda hace de Struts 2 el marco de trabajo más atractivo y potente para la creación de aplicaciones Web con arquitectura MVC.

3.3.1.2. Componentes de Struts 2

Struts 2 sigue siendo fiel a la arquitectura MVC, si bien los roles que desempeñan algunos de los componentes tradicionales cambian con respecto a las versiones 1.x.

La figura muestra los principales componentes de Struts 2, su ubicación dentro de las distintas capas de la aplicación y la interacción entre los mismos. [11]

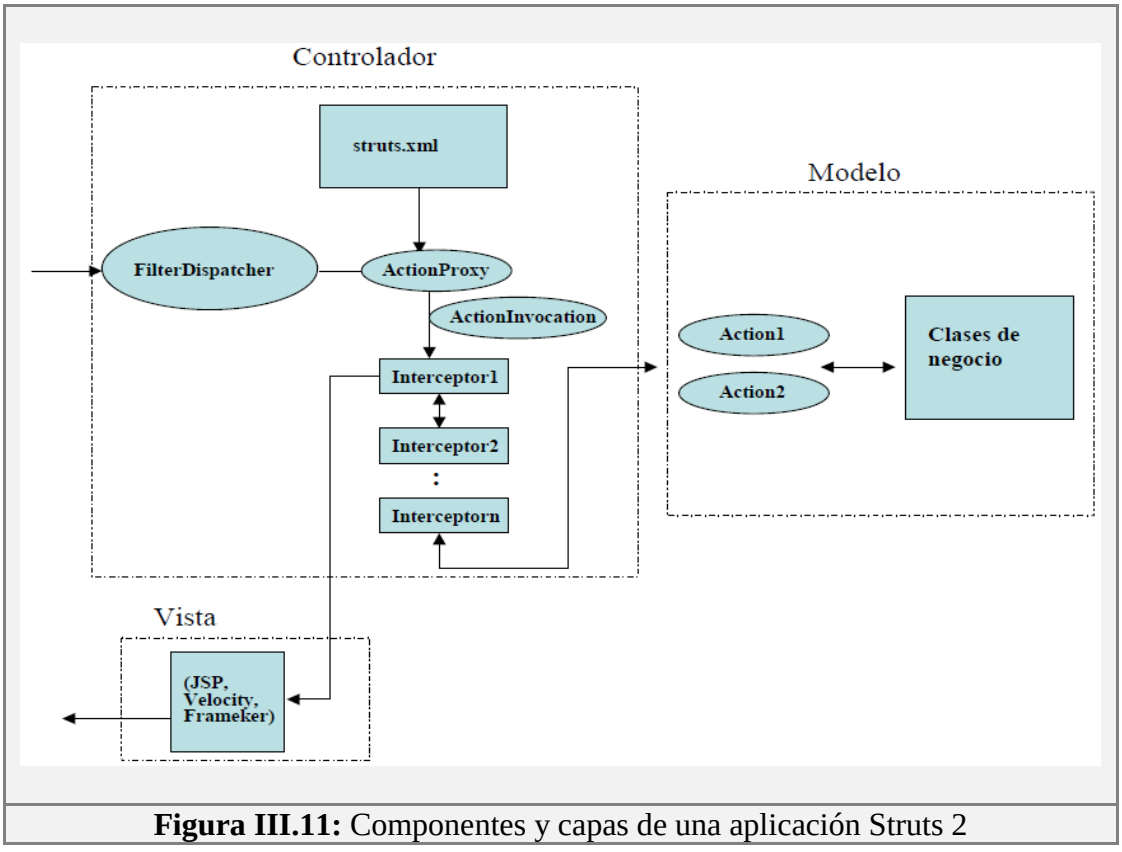


Figura III.11: Componentes y capas de una aplicación Struts 2

A continuación el funcionamiento y principales características de estos componentes, analizando su comportamiento a lo largo del ciclo de vida de la petición desde que ésta llega al Controlador hasta que se envía la respuesta al cliente.

[11] *Componentes de Struts 2*
http://es.scribd.com/doc/76999388/94/COMPONENTES-DE-STRUTS-2

3.3.1.2.1. FilterDispatcher

Este componente representa el punto de entrada a la aplicación, dirigiéndose a él todas las peticiones que llegan desde el cliente. FilterDispatcher hace en Struts 2 las veces de ActionServlet en Struts 1, analizando la petición recibida y determinando con el apoyo de otros objetos auxiliares y de la información almacenada en el archivo de configuración struts.xml el tipo de acción a ejecutar.

FilterDispatcher forma parte del API de Struts 2, concretamente, se incluye dentro del paquete org.apache.struts2.dispatcher.y, como se deduce de su propio nombre, es implementado mediante un filtro, por lo que debe ser registrado en el archivo web.xml de la aplicación.

3.3.1.2.2. Interceptores

Una vez determinada la acción a ejecutar, FilterDispatcher pasa el control de la petición a un objeto intermediario de tipo ActionProxy, éste crea un objeto ActionInvocation en el que almacena la información de la petición entrante, pasando a continuación el control de la misma a los interceptores.

Los interceptores constituyen una cadena de objetos que realizan una serie de tareas de pre-procesamiento previas antes de ejecutar la acción, como por ejemplo la validación de datos de usuario, rellenado de objetos con los campos de un formulario, etc., así como un post-procesamiento de los resultados una vez que éstos ha sido generados por las clases de acción. El funcionamiento de los interceptores es muy similar a los filtros servlet,

ejecutándose en cadena al recibir la petición, en el orden indicado en el archivo de configuración struts.xml y en orden inverso durante el envío de la respuesta al cliente.

El API de Struts incorpora un gran número de interceptores, pudiendo decidir el programador cuáles de ellos deben ejecutarse para cada acción simplemente indicándolo en el archivo de configuración struts.xml. Algunos de estos interceptores proporcionan a las clases de acción acceso a los distintos objetos del API Servlet, a fin de que éstas puedan controlar los datos manejados en la aplicación.

Un programador también puede implementar sus propios interceptores para realizar algún tipo de pre-procesamiento o post-procesamiento personalizado que no esté contemplado por ninguno de los interceptores del API. Para este propósito Struts 2 proporciona la interfaz Interceptor.

3.3.1.2.3. Action

Sin embargo, en Struts 2 este tipo de objetos presenta notables diferencias respecto a las versiones anteriores; se analiza las más significativas:

- ✓ Los objetos Action forman parte del modelo. La lógica de negocio se suele aislar en clases independientes o EJB, incluyéndose en las clases de acción las llamadas a los métodos expuestos por estos objetos.
- ✓ Implementación como clases POJO (Plain Old Java Objects)**.** La principal característica de las clases de acción de Struts 2 es que no tienen que heredar ni implementar ninguna clase o interfaz del API. Son clases estándares Java y cuyo único requerimiento es tener que proporcionar un método, que por convenio es llamado execute() al igual que las

clases de acción utilizadas en versiones anteriores del framework, en el que se deberán incluir las instrucciones a ejecutar para el procesamiento de la acción. Este método será invocado por el último interceptor de la cadena.

- ✓ Inclusión de métodos set/get. Otra de las características de Struts 2 es que se ha eliminado la utilización de clases de tipo ActionForm para la encapsulación de datos procedentes de un formulario cliente. En su defecto, estos datos son capturados por el propio objeto de acción invocado desde el Controlador, operación que es realizada con ayuda de uno de los interceptores incluidos en el API de Struts 2, resultando así transparente para el programador. Lo único que en este sentido habrá que codificar en las clases de acción (además del método execute() para el tratamiento de la petición) serán los datos miembro para el almacenamiento de los campos con sus correspondientes métodos set/get, de forma similar a como se hacía en las clases ActionForm de Struts 1.

3.3.1.2.4. Result

Este mecanismo permite que las acciones puedan retornar distintos tipos de resultados. Struts2 incluye una gran colección de resultados predefinidos que proporcionan por ejemplo el renderizado de una vista con JSP, FreeMarker o Velocity , encadenamiento de acciones, retornar streams, generar XML, etc.

Los tipos de resultado permiten la integración de diferentes herramientas de presentación. Es extensible y existen diferentes plugins que proporcionan diferentes tipos de resultado.

3.3.1.3. Ciclo de vida de una petición en Struts2

A continuación se ve los componentes por los que se va tratando la petición así como el tratamiento que se realiza según el orden de ejecución. El flujo comienza en el contenedor de servlets donde se recibe una petición HttpServletRequest y se pasa a través de la cadena de filtros standard del contenedor web para tratarla.

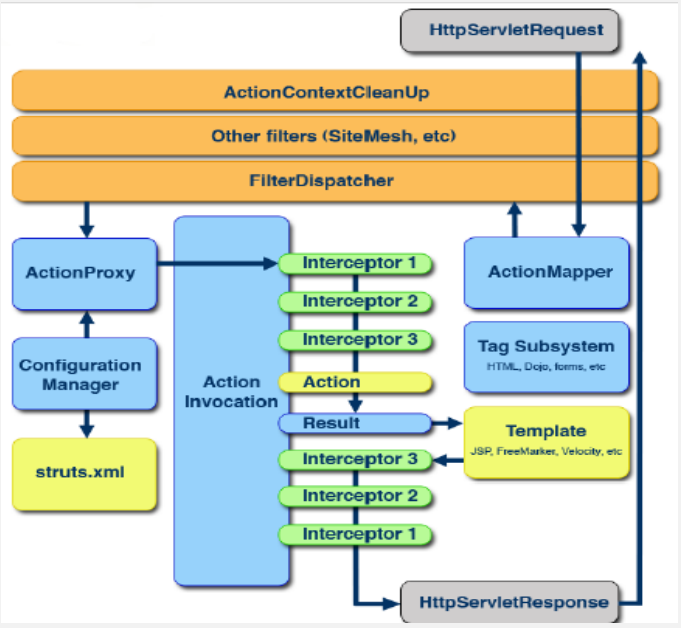


Figura III.12: Ciclo de vida de una petición en Struts 2

A continuación pasa por los siguientes componentes:

- ✓ **Filtro ActionContextCleanup**: Este es un filtro opcional que es útil para algunas ocasiones como la integración con otras tecnologías como puede ser SiteMesh.
- ✓ **Otros filtros**: Diversos filtros estándar del contenedor web que se pueden añadir y configurar para integrar diferentes tecnologías

- ✓ **Filtro FilterDispatcher**: Este filtro hace de *Front Controller* en Struts. Lo primero que hace este filtro es utilizar el *ActionMapper* para determinar si hay que invocar una acción o no. En caso de que haya que invocar una, el *FilterDispatcher* delega el control al *ActionProxy.*
- ✓ **ActionProxy:** El ActionProxy se ayuda de un ConfigurationManager (inicializado a partir del fichero de configuración struts.xml) para crear un ActionInvocation que implementa el patrón de diseño Chain of Responsability y Command El ActionInvocation se crea con una cadena de interceptores que se ejecutan envolviendo el método de ejecución de la acción. Esta cadena de interceptores se configura en el fichero struts.xml. El desarrollador puede crear nuevos interceptores y configurar diferentes cadenas de interceptores a ejecutar (denominadas *stacks*) tanto a nivel global, como a grupos o paquetes de acciones así como asignarlas a acciones individualmente.
- ✓ **ActionInvocation**: El ActionInvocation ejecuta los interceptores (si se ha configurado alguno) y después invoca a la acción. La acción se ejecuta y responde con una cadena indicando un nombre lógico de resultado. A continuación el ActionInvocation busca un resultado Result correspondiente a ese nombre lógico y lo ejecuta, provocando el renderizado de la JSP, plantillas o cualquier tipo de resultado que se haya integrado en el HttpServletResponse.

Para finalizar la cadena de interceptores sigue con su ejecución en orden inverso y la respuesta HTTPServletResponse pasa de nuevo por los filtros estándar finalizando así el tratamiento de la petición. [12]

---

[12] *Diseño e implementación de un framework de presentación para J2EE http://openaccess.uoc.edu/webapps/o2/bitstream/10609/6981/1/jcirianoTFC_memoria.pdf*

3.3.2. Spring MVC.

Figura III.13 : Logotipo de Spring MVC

3.3.2.1. Introducción

El Spring Framework cuenta con su propia MVC framework de aplicaciones web , que no estaba previsto inicialmente. Los desarrolladores de Spring decidieron escribir su propio framework de la red como una reacción a lo que percibían como el mal diseño y las deficiencias de otros frameworks disponibles. En particular, sentía que había una separación insuficiente entre las capas de presentación y solicitud de manipulación, y entre la capa de tratamiento de la petición y el modelo.

Spring MVC es un framework basado en peticiones cuyo objetivo de cada interfaz debe ser simple y clara para que sea fácil para los usuarios de Spring MVC para escribir sus propias implementaciones si así lo desean. MVC allana el camino para el código más limpio front-end. [13]

3.3.2.2. Componentes de Spring MVC

En el framework Web MVC cada DispatcherServlet tiene asociado su propio WebApplicactionContext, el cuál hereda todos los beans definidos en el WebApplicationContext raíz que pueden ser sobreescritos.

---

[13] *Spring Framework*
http://en.wikipedia.org/wiki/Spring_Framework

Una vez inicializado el *DispatcherServlet*, el framework busca un fichero llamado *[servlet-name]-servlet.xml* en el directorio WEB-INF de tu aplicación web y crea los beans allí definidos (si hay beans con el mismo nombre que los del contexto raíz los sobreescribe). En el ejemplo anterior se debe tener un fichero llamado *buscar-servlet.xml*.

El *DispatcherServlet* de Spring usa beans especiales para procesar las peticiones y mostrar las vistas apropiadas. Estos beans son parte del Framework y pueden ser configurados en el contexto *WebApplicationContext* como harías con otros beans. En la mayoría de los beans se proporcionan parámetros por defecto y no necesitan configuración. [14]

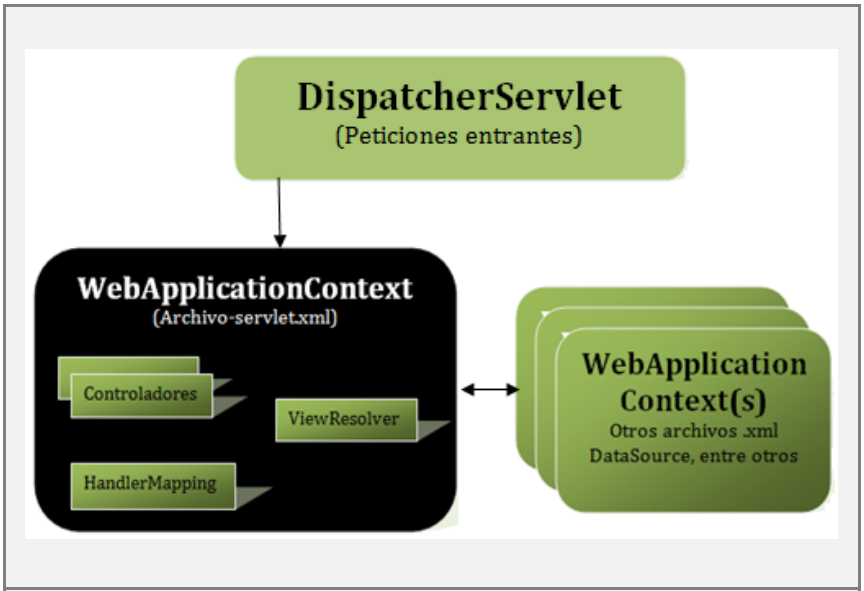


Figura III.14 : Arquitectura básica de Spring MVC

El DispatcherServlet clase es el controlador frontal de la estructura y es responsable de la delegación de control para las distintas interfaces durante las fases de ejecución de una petición HTTP en pocas palabras despacha las peticiones que recibe al controlador responsable de tratarlas.

[14] *Introducción a Aplicaciones REST con Spring 3.0 Web MVC Framework*
http://joseantoniosaiz.es/blog/introduccion-aplicaciones-rest-con-spring-3-0-web-mvc-framework/

Configurar el /WEB-INF/web.xml de la aplicación para que el dispatcher reciba las peticiones. Lee su configuración del fichero dispatcher-servlet.xml:

- ✓ Ubicado en el /WEB-INF/ de la aplicación
- ✓ Referencia al HandlerMapping
- ✓ Referencias a los controladores
- ✓ Referencia al ViewResolver

Las interfaces más importantes definidos por Spring MVC, y sus responsabilidades, se enumeran a continuación:

- ✓ *HandlerMapping* : Permite la selección de los objetos que manejan las solicitudes de entrada (los controladores) sobre la base de cualquier atributo o condición interna o externa a esas solicitudes.
- ✓ *HandlerAdapter* : Ayuda a la ejecución de los objetos que manejan las solicitudes de entrada. HandlerAdapter se utiliza para permitir que el DispatcherServlet ser indefinidamente extensible. El DispatcherServlet accede a todos los controladores instalados a través de este, lo que significa que no contiene el código específico de un tipo de controlador.
- ✓ *Controlador* : Se interpone entre Modelo y Vista para gestionar las solicitudes de entrada y redirigir a la respuesta correcta. Actúa como una puerta que dirige la información entrante. Cambia entre ir en el modelo o la vista.
- ✓ *ViewResolver* : Selecciona una vista sobre la base de un nombre lógico para el punto de vista (uso no es estrictamente necesario)

- ✓ *Ver* : Responsable de devolver una respuesta para el cliente. Algunas solicitudes pueden ir directamente a ver sin ir a la parte del modelo, mientras que otros pueden pasar por los tres.
- ✓ *HandlerInterceptor* : La interceptación de las solicitudes de entrada comparables, pero no es igual a Servlet filtros (el uso es opcional y no está controlada por *DispatcherServlet).*
- ✓ *LocaleResolver* : Resolver y, opcionalmente, el ahorro de la localidad de un usuario individual
- ✓ *MultipartResolver* : Facilitar el trabajo con la carga de archivos, envolviendo las solicitudes de entrada.

Las abstracciones ofrecidas por estas interfaces son de gran alcance, de modo de permitir una serie de variaciones en sus implementaciones, Spring MVC barcos con las implementaciones de todas estas interfaces y en conjunto ofrece un conjunto de características en la parte superior de la API Servlet. Spring MVC utiliza Java java.util.Map interfaz como una abstracción de datos orientada para el *modelo* donde las claves se espera que sean los valores de cadena.

La facilidad de probar las implementaciones de estas interfaces parece una ventaja importante del alto nivel de abstracción que ofrece Spring MVC. *DispatcherServlet* está estrechamente ligado a Spring de Inversión de control de contenedores para la configuración de las capas de aplicaciones web. [15]

---

[15] *Spring Framework*
http://en.wikipedia.org/wiki/Spring_Framework

3.3.2.3. Ciclo de vida de una petición en Spring MVC

Para comprender la arquitectura se muestra el ciclo de vida de una petición o request en Spring MVC, se tienen los siguientes pasos.

1. El navegador manda un request y lo recibe un DispatcherServlet.
2. Se debe escoger que Controller manejará el request, para esto el HandlerMapping mapea los diferentes patrones de URL hacia los controladores, y se le regresa al DispatcherServlet el Controller elegido.
3. El Controller elegido toma el request y ejecuta la tarea.
4. El Controller regresa un ModelAndView al DispatcherServlet.
5. Si el ModelAndView contiene un nombre lógico de un View se tiene que utilizar un ViewResolver para buscar ese objeto View que representara el request modificado.
6. Finalmente el DispacherServlet despacha el request al View.

Spring cuenta con una gran cantidad de controladores de los cuales se puede elegir dependiendo de la tarea. [16]

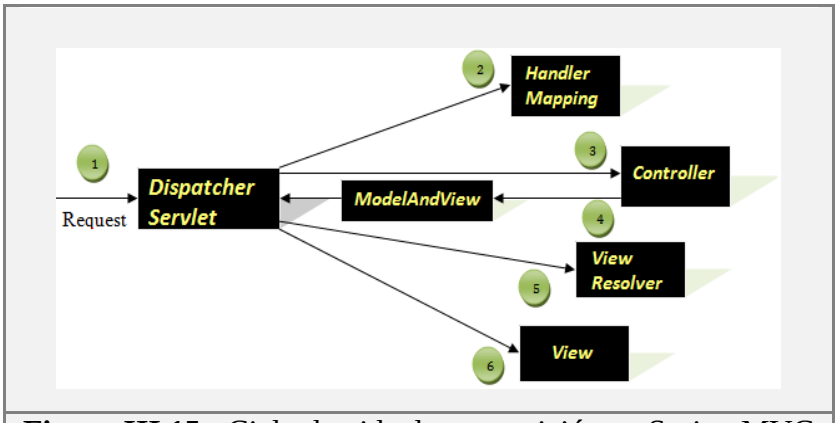


Figura III.15 : Ciclo de vida de una petición en Spring MVC

[16] *Spring un framework de aplicación http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf*

3.4. Análisis Comparativo de Frameworks

3.4.1. Determinación de los parámetros de evaluación de los frameworks

A continuación se definen los parámetros que permiten evaluar los frameworks Struts2 y Spring MVC los mismos que tendrán su respectiva apreciación según los valores definidos anteriormente.

3.4.1.1. Selección de los parámetros para el análisis de los frameworks

TABLA III.XIII

Parámetros a evaluar

Parámetros	Variable	Indicadores
1. Patrón MVC	Modelo	✓ Compatibilidad con bases de datos ✓ Manejo de conexión a una base de datos ✓ Testeabilidad ✓ Interacción con la base de datos.
	Vista	✓ Generación de interfaces ✓ Manejo de estilos ✓ Sistema de plantillas ✓ Vinculación entre datos
	Controlador	✓ Navegabilidad entre páginas ✓ Generación de etiquetas de formulario ✓ Validaciones de formulario ✓ Manejo de sesiones

2. Otras características	Otros	✓ Excepciones. ✓ Internalización
3. Tiempo	Tiempo de Codificación	✓ Facilidad para entender el código ✓ Número de líneas de código
	Tiempo de aprendizaje	✓ Facilidad de aprendizaje ✓ Comprensibilidad
	Tiempo de desarrollo	✓ Instalación y configuración ✓ Implementación de base de datos ✓ Codificación de la capa modelo ✓ Codificación de la capa vista ✓ Codificación de la capa controlador ✓ Pruebas

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.1.2. Determinación de la importancia de los parámetros.

TABLA III.XIV

Importancia de parámetros

Parámetro	Valor
Patrón MVC	30%
Otras características	20%

Tiempo	50%

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2. Comparación de los frameworks a evaluar

3.4.2.1. Patrón MVC

3.4.2.1.1. Modelo

3.4.2.1.1.1. Análisis de los indicadores del Modelo del Patrón MVC

####### 3.4.2.1.1.1.1. Compatibilidad de base de datos

Indicador que permite evaluar la compatibilidad con distintas bases de datos debido a la existencia de una gran cantidad de gestores de base de datos.

TABLA III.XV

Comparación del indicador Compatibilidad de base de datos

FRAMEWORK	COMPATIBILIDAD DE BASE DE DATOS
Struts 2	Struts 2 es compatible con la mayoría de los gestores de base de datos como: MySQL, Oracle, SQL Server, etc.
Spring MVC	Spring Mvc brinda compatibilidad a múltiples motores de datos de una forma muy sencilla entre ellos se tiene Postgres, MySQL, Oracle, etc.

Elaborado por: Ximena Moposita, Mercedes Morán

####### 3.4.2.1.1.1.2. Manejo de conexión a una base de datos

Este indicador evalúa la sencillez y la facilidad que tiene el framework al momento de realizar la configuración de la conexión a una base de datos única.

TABLA III.XVI

Comparación del indicador Manejo de conexión a una base de datos

FRAMEWORK	MANEJO DE CONEXIÓN A UNA BASE DE DATOS
Struts 2	La configuración de conexión de una base de datos con Struts 2 se lo hace a través de JDBC como en cualquier clase de J2SE.
Spring MVC	Mediante dos opciones se hace el acceso, conexión y manejo de base de datos que depende del desarrollador basándose en la complejidad de la aplicación, si se trata de una aplicación sencilla se puede hacer mediante JDBC, o si es una aplicación más robusta se hace el uso del modulo de Spring ORM.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.1.1.3. Testeabilidad

Con este indicador se analiza si los frameworks evaluados, tiene la capacidad de realizar un test unitario sin necesidad de iniciar un contenedor.

TABLA III.XVII

Comparación del indicador Testeabilidad

FRAMEWORK	TESTEABILIDAD
Struts 2	StrutsTestCase es una extensión de junit que permite realizar las pruebas de dos maneras diferentes: mediante mock objects que no requiere la ejecución dentro de un contenedor de aplicaciones, o mediante el proyecto Cactus para la ejecución dentro de un contenedor (un contender de servlets ligero).

Spring MVC	Los test de unidad se permiten mediante la adaptación con JUnit, acceso al factory de beans directamente desde los test unitarios (unit tests) y los mock objects.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.1.1.4. Interacción con la base de datos

Se aprecia la gestión de la aplicación con la base de datos, permitiendo analizar las bondades que se presentan al ejecutar consultas a la base de datos.

TABLA III.XVIII

Comparación de la Interacción con la base de datos

FRAMEWORK	INTERACCIÓN CON LA BASE DE DATOS
Struts 2	En el modelo que es la parte responsable de la gestión de la información, hace referencia a los datos que maneja la aplicación y las reglas de negocio que operan sobre ellos y que se traducen en Struts 2 en las acciones. En este punto se suelen incluir aquellas clases, herramientas, librerías, etc., que permiten el acceso a los datos, así como el modelado de los mismos.
Spring MVC	En el modelo se realiza toda la lógica de negocio de la aplicación, es decir, aquí hay una capa que se llama capa de servicios, donde se hace uso de la base de datos.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.1.2. Resultados

TABLA III.XIX

Resultados de la variable Modelo

Frameworks / **Indicadores**	Struts 2	Spring MVC
Compatibilidad de base de datos	4	4
Manejo de conexión a una base de datos	3	4
Testeabilidad	4	4
Interacción con la base de datos	2	2
Total	**13**	**14**

Elaborado por: Ximena Moposita, Mercedes Morán

$$\mathbf{C_{Struts}} = \sum_{i=1}^{4} \text{VPEi} = 4 + 3 + 4 + 4 = 15$$

$$\mathbf{C_{SpringMVC}} = \sum_{i=1}^{4} \text{VPEi} = 4 + 4 + 4 + 4 = 16$$

3.4.2.1.1.3. Interpretación

Spring MVC y Struts 2 tienen una magnifica compatibilidad con los diferentes gestores de base de datos tanto libres como propietarios permitiendo utilizarlos dependiendo a las necesidades de cada proyecto.

Spring MVC presenta una ligera ventaja sobre Struts 2 en cuanto al manejo de conexión de base de datos ya que este permite integrarse a alguna herramienta ORM directamente al API del mismo, dichas configuraciones son sencillas e intuitivas.

A la hora de realizar test unitarios ambos frameworks son estupendos ya que Spring MVC está basado en POJOs y en Struts 2 las acciones engloban la lógica de negocios y los JavaBeans, admitiendo de esta manera que los test unitarios sean más sencillos elaborarlos.

3.4.2.1.2. Vista

3.4.2.1.2.1. Análisis de los indicadores de la Vista del Patrón MVC

3.4.2.1.2.1.1. Generación de interfaces

La generación de interfaces que presentan los frameworks para las tareas más comunes de una aplicación web.

TABLA III.XX

Comparación del indicador Generación de interfaces

FRAMEWORK	GENERACIÓN DE INTERFACES
Struts 2	La posibilidad de reutilizar código proporciona una gran flexibilidad en el diseño de las mismas y ofrece una gran variedad de posibilidades a los desarrolladores.
Spring MVC	Spring está diseñado con interfaces para que el desarrollador pueda utilizarlas, promoviendo así la reutilización de código y un estándar del paradigma orientado a objetos. Una ventaja práctica de Spring, para intentar hacer código menos repetitivo al momento de hacer una lectura y/o escritura de datos en una base de datos, es que proporciona una serie de clases e interfaces para simplificar estas acciones.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.2.1.2. Manejo de estilos

Este indicador permite analizar la capacidad y la flexibilidad que tiene el framework para asociar las paginas a hojas de estilo.

TABLA III.XXI

Comparación del indicador Manejo de estilos

FRAMEWORK	MANEJO DE ESTILOS
Struts 2	Con la etiqueta auxiliar Head de struts 2 que se coloca dentro de la etiqueta head de HTML y se encarga de generar distintos elementos necesarios para otras etiquetas, como las etiquetas para la carga de hojas de estilo o scripts. org.apache.struts2.views.jsp.ui.HeadTag
Spring MVC	El uso de hojas de estilo no interviene en la creación de páginas web utilizando el framework Spring MVC. Se lo utiliza en la cabecera de la página web.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.2.1.3. Sistema de plantillas

Este indicador hace referencia a los mecanismos que utilizan los frameworks para la decoración de páginas.

TABLA III.XXII

Comparación del indicador Sistema de plantillas

FRAMEWORK	SISTEMA DE PLANTILLAS
Struts 2	Se puede utilizar junto con Tiles, pero requiere configuración en cada página, además se puede utilizar junto con SiteMesh, el cual presenta

	una instalación y posterior uso muchos más sencillos.
Spring MVC	Admite seleccionar JSP, Velocity, Tiles, iText o POI permitiendo intégralos con el resto de la aplicación.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.2.1.4. Vinculación entre datos.

Se analiza cómo se maneja el enlace de datos entre entre las páginas Html y los objetos Java.

TABLA III.XXIII

Comparación del indicador Vinculación entre datos

FRAMEWORK	VINCULACIÓN ENTRE DATOS
Struts 2	En Struts 2, ActionContext contiene todos los datos asociados con el procesamiento de una solicitud, el mismo que ayuda a mover los datos de la solicitud a la ValueStack que permite la vinculación de los datos. El lenguaje de expresión OGNL hace el trabajo de convertir los datos de cadena a base que sus correspondientes tipos de Java, una vez que genera los resultados convierte los tipos de Java de la propiedad en el ValueStack en la salida HTML basada en cadena.
Spring MVC	Se utiliza principalmente en Spring MVC el DataBinder para ligar los objetos que se utiliza para procesar la entrada del usuario. El enlace de datos se configura en la clase WebDataBinder, este proceso de unión se puede personalizar mediante la especificación de

	campos permitidos, campos requeridos, y los editores personalizados. Primavera inyecta una instancia de esta clase en cualquier método de control que ha sido anotado con @InitBinder. Este objeto se utiliza para definir las normas de enlace de datos para el controlador.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.2.2. Resultados

TABLA III.XXIV

Resultados de la variable Vista

Frameworks / **Indicadores**	Struts 2	Spring MVC
Generación de interfaces	3	3
Manejo de estilos	4	4
Sistema de plantillas	4	4
Vinculación de datos	4	3
Total	**15**	**14**

Elaborado por: Ximena Moposita, Mercedes Morán

$$\mathbf{C_{Struts}} = \sum_{i=1}^{4} \text{VPEi} = 3 + 4 + 4 + 4 = 15$$

$$\mathbf{C_{SpringMVC}} = \sum_{i=1}^{4} \text{VPEi} = 3 + 4 + 4 + 3 = 14$$

3.4.2.1.2.3. Interpretación

Los frameworks poseen interfaces sencillas y con diseños no muy elaborados. Además poseen características similares para el manejo de estilos y plantillas para la creación de interfaces refinadas.

La vinculación de datos en Struts 2 es ligeramente beneficioso por el uso de OGNL que es mucho más rico que el Spring MVC que le permite hacer cosas que no son posible como la de simplificar la implementación de aplicaciones.

3.4.2.1.3. Controlador

3.4.2.1.3.1. Análisis de los indicadores del Controlador del Patrón MVC

3.4.2.1.3.1.1. Navegabilidad entre páginas

El indicador permite analizar como el framework maneja la redirección y el enrutamiento del enlace entre páginas de la aplicación.

TABLA III.XXV

Comparación del indicador Navegabilidad entre páginas

FRAMEWORK	NAVEGABILIDAD ENTRE PAGINAS
Struts 2	En Struts 2 la etiqueta < result\ > permite implementar 2 tipos de re direccionamiento. *El tipo dispatcher* que permite redirigir la acción a una página JSP de resultado, no se cambia la URL del navegador y se conserva los parámetros en la consulta, no permite realizar una redirección a un

	recurso externo o una URL absoluta. Además es el más usado. *El tipo Redirect* este tipo permite realizar una redirección completa a otra URL, cambia la URL en el navegador y se pierden los parámetros presentes en la consulta, permite realizar redirección a recursos internos o externos, es rápida y se usa para administrar el doble envió.
Spring MVC	Spring MVC permite realizar redirecciones a un manejador retornando un objeto de tipo RedirectView o utilizando el prefijo redirect en el nombre de la vista además utiliza el prefijo forward que permite redirigir de una página a otra, pero esta vez sin intervención del cliente, de forma que la URL aparentemente sea la misma que la original solicitada.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.3.1.2. Generación de etiquetas UI

Analiza en cada framework si se tiene la capacidad de crear los formularios mediante la generación automática de etiquetas de interfaz de usuario.

TABLA III.XXVI

Comparación del indicador Generación de etiquetas UI

FRAMEWORK	GENERACIÓN DE ETIQUETAS UI
Struts 2	Las etiquetas de interfaz de usuario son simples y fáciles de usar. No se necesita escribir ningún código HTML, las etiquetas de interfaz de

	usuario automáticamente se generan basándose en el tema que se elija.
Spring MVC	Brinda una serie de etiquetas para la creación de los formularios llevar a cabo las tareas más fácilmente.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.3.1.3. Validaciones de formulario

Indicador que permite analizar si el framework posee seguridad validando sus datos en el momento en que se produzcan entradas desde el exterior del sistema.

TABLA III.XXVII

Comparación del indicador Validaciones de formulario

FRAMEWORK	VALIDACIONES DE FORMULARIO
Struts 2	Este framework proporciona varias maneras de realizar las validaciones de forma automática, se la realiza mediante dos interceptores. ValidationInterceptor que crea una lista de errores específicos para cada uno de los campos que no pase la validación. Y el interceptor "workflow" que verifica si hay errores de validación. Para utilizar cualquiera de los validadores predefinidos, no se necesita configuración inicial. Struts 2 tiene tres formas de realizar validaciones, mediante un archivo XML, anotaciones y manual.
Spring MVC	En SpringMVC los validadores no dependen de la API de Servlets, incluso cuando se utiliza commons-validator. Esto permite a los

	validadores ser reutilizados. Las anotaciones se utilizan para lo que especifican los argumentos Validator y se debe aplicar a cada campo. Spring MVC es compatible con el estándar de Java para la validación Bean JSR 303 que respalda la validación de un formulario HTML. Las entradas de un formulario se validan automáticamente.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.3.1.4. Manejo de sesiones

Se analiza los diferentes enfoques que se pueden llevar a cabo a la hora de manejar las sesiones en los frameworks.

TABLA III.XXVIII

Comparación del indicador Manejo de sesiones

FRAMEWORK	MANEJO DE SESIONES
Struts 2	*SessionAware*, interfaz que permite que los Actions reciban cierta información al momento de inicializarlas. Esta interfaz es relevante sólo si la acción se utiliza en un entorno de servlet. *Uso del objeto "ActionContext",* que es el contexto en el que una Action se ejecuta. Cada contexto es básicamente un contenedor de objetos que una acción necesita para su ejecución, como el período de sesiones.
Spring MVC	*HttpSession*, interfaz que es implementada de una manera simple, pero no es recomendable si se quiere llevar a cabo test unit de su contenedor

	Alcance del controlador, se puede crear una instancia del objeto que desea almacenar en el ámbito de la sesión como una variable del controlador, con lo que se logra que este muy limpio y permita realizar test sin dificultad. Pero para cada sesión se crea un nuevo controlador provocando problemas de replicación en sistemas de gran escala. *Alcance de los objetos en la sesión*, se trata de una relación del ámbito de la sesión, con el objeto que se quiere almacenar en la sesión. Para cada solicitud, se crea una instancia del controlador lo que resultaría un problema de escalabilidad pero se mantienen solo los datos de sesión pertinentes. *Utilizar <aop:scoped-proxy/>*, los datos de la sesión se declaran como un bean de Spring regular usando esta etiqueta, permitiendo que sólo los datos de la sesión se almacena en la HttpSession, el controlador es una única instancia y admite a realizar test unitarios sin dificultad. Este método es el más difícil de entender.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.1.3.2. Resultados

TABLA III.XXIX

Resultados de la variable Controlador

Frameworks / **Indicadores**	Struts 2	Spring MVC
Navegabilidad entre paginas	3	4

Generación de etiquetas de formulario	4	4
Validación de formularios	4	4
Manejo de sesiones	3	4
Total	**14**	**16**

Elaborado por: Ximena Moposita, Mercedes Morán

$$\mathbf{C_{Struts}} = \sum_{i=1}^{4} \text{VPEi} = 3 + 4 + 4 + 3 = 14$$

$$\mathbf{C_{SpringMVC}} = \sum_{i=1}^{4} \text{VPEi} = 4 + 4 + 4 + 4 = 16$$

3.4.2.1.3.3. Interpretación

Struts 2 y Spring MVC en su análisis muestran resultados casi semejantes. En la navegabilidad entre páginas los frameworks proporcionan un conjunto de recursos para la navegación diseñados para conseguir un resultado óptimo.

Estos frameworks en la generación de etiquetas de formulario insertan componentes para manipular las propiedades de los elementos interactivos que se presentan en los formularios de una forma sencilla y amigable.

Al momento en que el usuario ingresa datos es muy importante que se validen cada uno de los campos, tarea que realizan los frameworks de una manera exitosa ya que permiten que

el programador no tenga que escribir una sola instrucción de código Java y aseguran que los datos que se introducen son los correctos.

En cuanto a sesiones Spring MVC hace un mejor seguimiento de un usuario a través de la aplicación que Struts 2 porque tiene un mejor alcance del controlador.

3.4.2.2. Otras características

3.4.2.2.1. Análisis de los indicadores de otras características de los framework

3.4.2.2.1.1. Excepciones

Se analiza los mecanismos que posee cada framework al momento de manejar las una situación que puede provocar un fallo en el programa (excepciones).

TABLA III.XXX

Comparación del indicador Excepciones

FRAMEWORK	EXCEPCIONES
Struts 2	Struts 2 proporciona un mecanismo de manejo de excepciones declarativa que se puede configurar a nivel mundial o para una acción específica. . Esta capacidad puede reducir la cantidad de código de manejo de excepciones dentro de las acciones necesarias en determinadas circunstancias.
Spring MVC	Spring MVC utiliza el SimpleMappingExceptionResolver que es una aplicación de HandlerExceptionResolver que permite definir las prioridades de excepción, así como el envío de notificaciones diferentes en función de las prioridades además admite excepciones

	específicas.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.2.1.2. Internacionalización

Indicador que analiza si el framework utiliza un medio para que pueda ser adaptada la aplicación web a distintos idiomas y regiones sin necesidad de cambios de ingeniería.

TABLA III.XXXI

Comparación del indicador Internacionalización

FRAMEWORK	INTERNACIONALIZACIÓN
Struts 2	Por defecto su función es meter un objeto Locale en sesión con el valor del sistema del cliente o el valor que se le pasa en el parámetro request_locale. Permitiendo utilizar un ResourceBundle único por cada locale. Permite definir archivos de mensajes para diferentes idiomas haciendo un trabajo fácil en Struts 2.
Spring MVC	Spring MVC, viene con unos cuantos **"LocaleResolver"** para apoyar la internacionalización o las características de múltiples idiomas. Spring MVC compatible con el estándar de Java paquete de recursos para acceder a específicos de la localidad de mensajes de texto en los archivos de propiedad.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.2.2. Resultados

TABLA III.XXXII

Resultados de la variable Otras características

Frameworks / Indicadores	Struts 2	Spring MVC
Excepciones	3	4
Internacionalización	4	4
Total	**7**	**8**

Elaborado por: Ximena Moposita, Mercedes Morán

$$C_{Struts} = \sum_{i=1}^{2} VPEi = 3 + 4 = 7$$

$$C_{SpringMVC} = \sum_{i=1}^{2} VPEi = 4 + 4 = 8$$

3.4.2.2.3. Interpretación

En el manejo de excepciones Spring MVC es superior que Struts 2 porque hace un mejor envió en las notificaciones de problemas que se presentan en la ejecución de la aplicación, además mapea los tipos de excepciones a las páginas de error, logrando manejar excepciones de una forma mucho más flexible. Java incorpora soporte para la internacionalización y estos dos frameworks hacen uso de esta característica fácilmente adaptando la aplicación a diferentes idiomas mediante la adición de componentes específicos de la localidad y la traducción de texto.

3.4.2.3. Tiempo

3.4.2.3.1. Codificación

3.4.2.3.1.1. Análisis de los indicadores de codificación

3.4.2.3.1.1.1. Facilidad para entender el código

Indicador que muestra el tiempo para entender cómo esta codificada una aplicación web usando el framework a base de la experiencia personal de las autoras.

TABLA III.XXXIII

Comparación del indicador Facilidad para entender el código

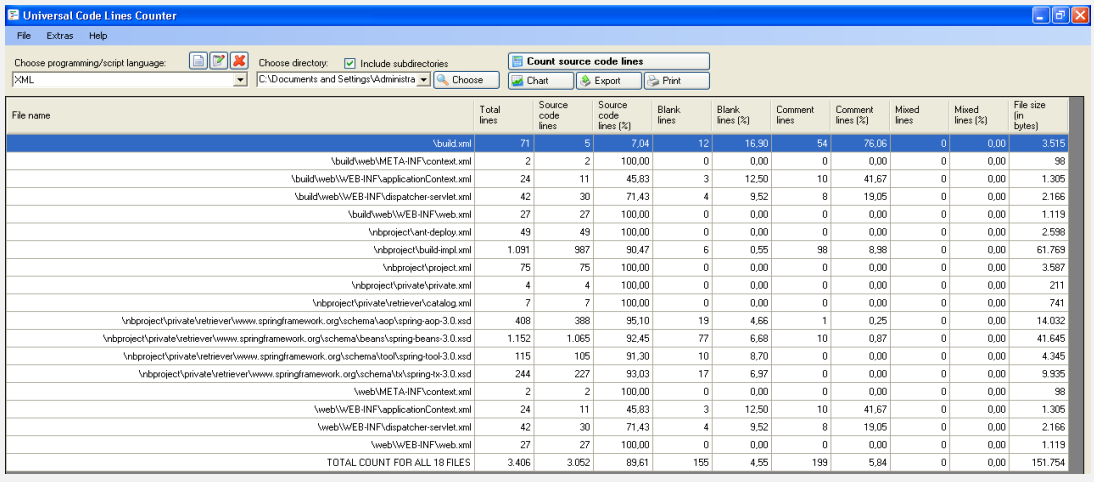
FRAMEWORK	FACILIDAD PARA ENTENDER EL CÓDIGO
Struts 2	Para entender la forma de codificación en este framework se tomo 15 días, del 16/04/2012 al 04/05/2012.
Spring MVC	Con el apoyo de los recursos disponibles se logro comprender la codificación de Spring MVC en 16 días, del 15/05/2012 al 05/06/2012.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.3.1.1.2. Número de líneas de código

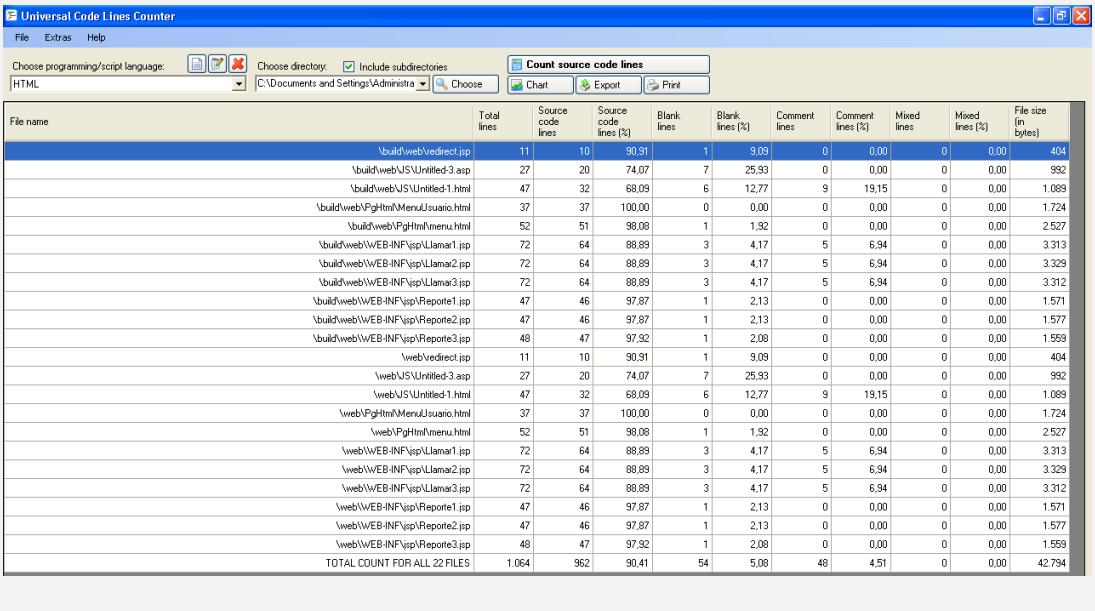
Este indicador analiza cuantas líneas de código contiene el modulo de prueba desarrollado con los frameworks. Los resultados se obtuvieron del programa Universal Code Lines Counter Version 1.1.5 mostrados por tipo de lenguaje.

Líneas de código del Modulo de prueba realizado con Spring MVC



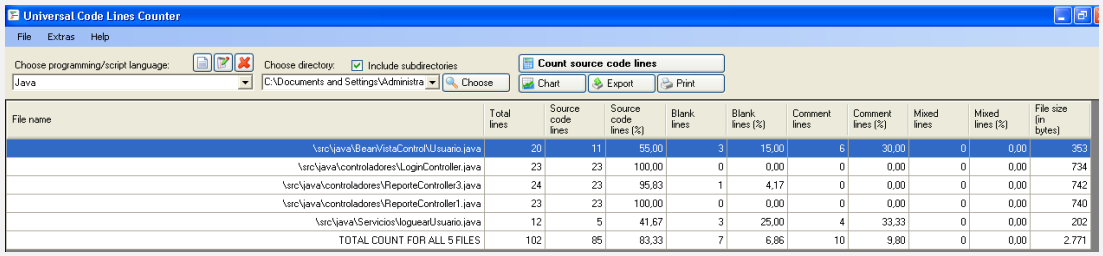
File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\build.xml	71	5	7,04	12	16,90	54	76,06	0	0,00	3.515
\build\web\META-INF\context.xml	2	2	100,00	0	0,00	0	0,00	0	0,00	98
\build\web\WEB-INF\applicationContext.xml	24	11	45,83	3	12,50	10	41,67	0	0,00	1.305
\build\web\WEB-INF\dispatcher-servlet.xml	42	30	71,43	4	9,52	8	19,05	0	0,00	2.166
\build\web\WEB-INF\web.xml	27	27	100,00	0	0,00	0	0,00	0	0,00	1.119
\nbproject\ant-deploy.xml	49	49	100,00	0	0,00	0	0,00	0	0,00	2.598
\nbproject\build-impl.xml	1.091	987	90,47	6	0,55	98	8,98	0	0,00	61.769
\nbproject\project.xml	75	75	100,00	0	0,00	0	0,00	0	0,00	3.587
\nbproject\private\private.xml	4	4	100,00	0	0,00	0	0,00	0	0,00	211
\nbproject\private\retriever\catalog.xml	7	7	100,00	0	0,00	0	0,00	0	0,00	741
\nbproject\private\retriever\www.springframework.org\schema\aop\spring-aop-3.0.xsd	408	388	95,10	19	4,66	1	0,25	0	0,00	14.032
\nbproject\private\retriever\www.springframework.org\schema\beans\spring-beans-3.0.xsd	1.152	1.065	92,45	77	6,68	10	0,87	0	0,00	41.645
\nbproject\private\retriever\www.springframework.org\schema\tool\spring-tool-3.0.xsd	115	105	91,30	10	8,70	0	0,00	0	0,00	4.345
\nbproject\private\retriever\www.springframework.org\schema\tx\spring-tx-3.0.xsd	244	227	93,03	17	6,97	0	0,00	0	0,00	9.935
\web\META-INF\context.xml	2	2	100,00	0	0,00	0	0,00	0	0,00	98
\web\WEB-INF\applicationContext.xml	24	11	45,83	3	12,50	10	41,67	0	0,00	1.305
\web\WEB-INF\dispatcher-servlet.xml	42	30	71,43	4	9,52	8	19,05	0	0,00	2.166
\web\WEB-INF\web.xml	27	27	100,00	0	0,00	0	0,00	0	0,00	1.119
TOTAL COUNT FOR ALL 18 FILES	3.406	3.052	89,61	155	4,55	199	5,84	0	0,00	151.754

Figura III.16: Líneas de código .xml en Spring MVC



File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\build\web\redirect.jsp	11	10	90,91	1	9,09	0	0,00	0	0,00	404
\build\web\JS\Untitled-3.asp	27	20	74,07	7	25,93	0	0,00	0	0,00	992
\build\web\JS\Untitled-1.html	47	32	68,09	6	12,77	9	19,15	0	0,00	1.089
\build\web\PgHtml\MenuUsuario.html	37	37	100,00	0	0,00	0	0,00	0	0,00	1.724
\build\web\PgHtml\menu.html	52	51	98,08	1	1,92	0	0,00	0	0,00	2.527
\build\web\WEB-INF\jsp\Llamar1.jsp	72	64	88,89	3	4,17	5	6,94	0	0,00	3.313
\build\web\WEB-INF\jsp\Llamar2.jsp	72	64	88,89	3	4,17	5	6,94	0	0,00	3.329
\build\web\WEB-INF\jsp\Llamar3.jsp	72	64	88,89	3	4,17	5	6,94	0	0,00	3.312
\build\web\WEB-INF\jsp\Reporte1.jsp	47	46	97,87	1	2,13	0	0,00	0	0,00	1.571
\build\web\WEB-INF\jsp\Reporte2.jsp	47	46	97,87	1	2,13	0	0,00	0	0,00	1.577
\build\web\WEB-INF\jsp\Reporte3.jsp	48	47	97,92	1	2,08	0	0,00	0	0,00	1.559
\web\redirect.jsp	11	10	90,91	1	9,09	0	0,00	0	0,00	404
\web\JS\Untitled-3.asp	27	20	74,07	7	25,93	0	0,00	0	0,00	992
\web\JS\Untitled-1.html	47	32	68,09	6	12,77	9	19,15	0	0,00	1.089
\web\PgHtml\MenuUsuario.html	37	37	100,00	0	0,00	0	0,00	0	0,00	1.724
\web\PgHtml\menu.html	52	51	98,08	1	1,92	0	0,00	0	0,00	2.527
\web\WEB-INF\jsp\Llamar1.jsp	72	64	88,89	3	4,17	5	6,94	0	0,00	3.313
\web\WEB-INF\jsp\Llamar2.jsp	72	64	88,89	3	4,17	5	6,94	0	0,00	3.329
\web\WEB-INF\jsp\Llamar3.jsp	72	64	88,89	3	4,17	5	6,94	0	0,00	3.312
\web\WEB-INF\jsp\Reporte1.jsp	47	46	97,87	1	2,13	0	0,00	0	0,00	1.571
\web\WEB-INF\jsp\Reporte2.jsp	47	46	97,87	1	2,13	0	0,00	0	0,00	1.577
\web\WEB-INF\jsp\Reporte3.jsp	48	47	97,92	1	2,08	0	0,00	0	0,00	1.559
TOTAL COUNT FOR ALL 22 FILES	1.064	962	90,41	54	5,08	48	4,51	0	0,00	42.794

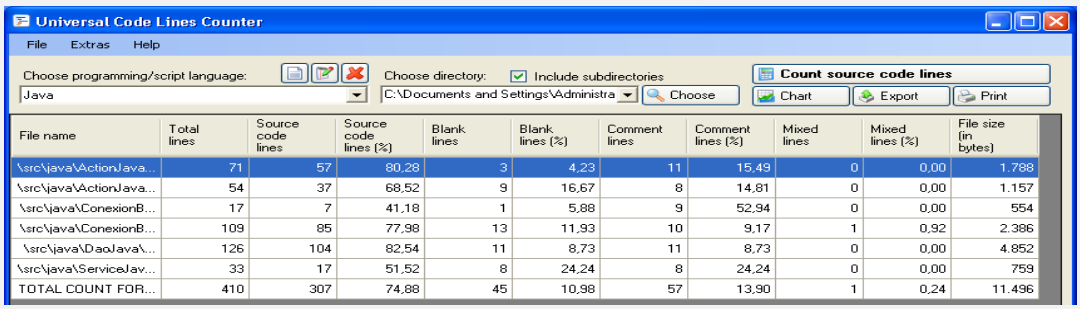
Figura III.17: Líneas de código .html y .jsp en Spring MVC



File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\src\java\BeanVistaControl\Usuario.java	20	11	55,00	3	15,00	6	30,00	0	0,00	353
\src\java\controladores\LoginController.java	23	23	100,00	0	0,00	0	0,00	0	0,00	734
\src\java\controladores\ReporteController3.java	24	23	95,83	1	4,17	0	0,00	0	0,00	742
\src\java\controladores\ReporteController1.java	23	23	100,00	0	0,00	0	0,00	0	0,00	740
\src\java\Servicios\logueatUsuario.java	12	5	41,67	3	25,00	4	33,33	0	0,00	202
TOTAL COUNT FOR ALL 5 FILES	102	85	83,33	7	6,86	10	9,80	0	0,00	2.771

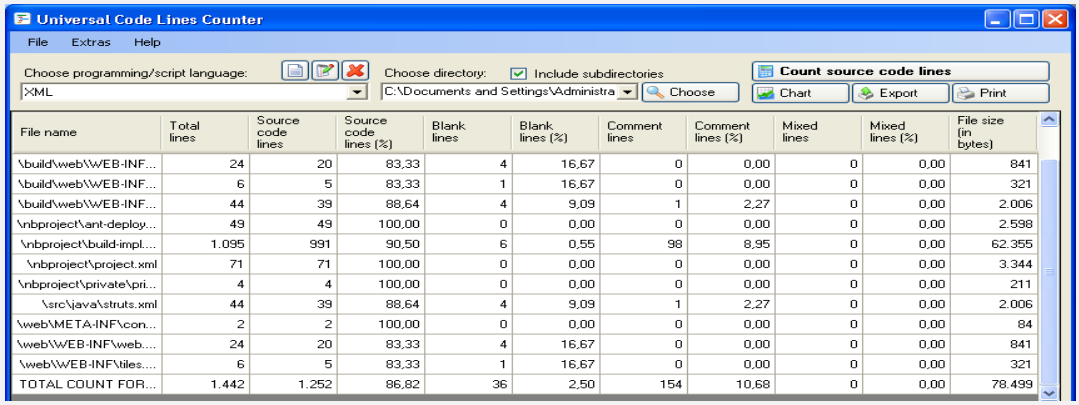
Figura III.18: Líneas de código .java en Spring MVC

Líneas de código del Modulo de prueba realizado con Struts2



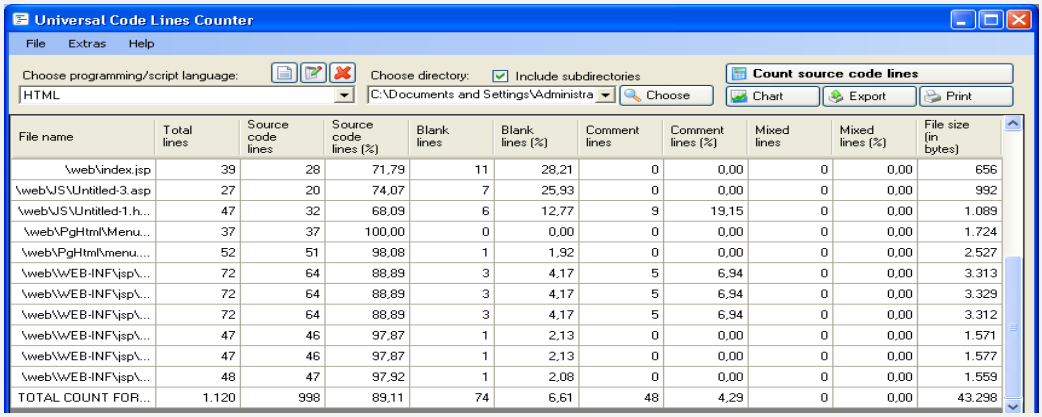
File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\src\java\ActionJava...	71	57	80,28	3	4,23	11	15,49	0	0,00	1.788
\src\java\ActionJava...	54	37	68,52	9	16,67	8	14,81	0	0,00	1.157
\src\java\ConexionB...	17	7	41,18	1	5,88	9	52,94	0	0,00	554
\src\java\ConexionB...	109	85	77,98	13	11,93	10	9,17	1	0,92	2.386
\src\java\DaoJava\...	126	104	82,54	11	8,73	11	8,73	0	0,00	4.852
\src\java\ServiceJav...	33	17	51,52	8	24,24	8	24,24	0	0,00	759
TOTAL COUNT FOR...	410	307	74,88	45	10,98	57	13,90	1	0,24	11.496

Figura III.19: Líneas de código .java en Struts 2



File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\build\web\WEB-INF...	24	20	83,33	4	16,67	0	0,00	0	0,00	841
\build\web\WEB-INF...	6	5	83,33	1	16,67	0	0,00	0	0,00	321
\build\web\WEB-INF...	44	39	88,64	4	9,09	1	2,27	0	0,00	2.006
\nbproject\ant-deploy...	49	49	100,00	0	0,00	0	0,00	0	0,00	2.598
\nbproject\build-impl....	1.095	991	90,50	6	0,55	98	8,95	0	0,00	62.355
\nbproject\project.xml	71	71	100,00	0	0,00	0	0,00	0	0,00	3.344
\nbproject\private\pri...	4	4	100,00	0	0,00	0	0,00	0	0,00	211
\src\java\struts.xml	44	39	88,64	4	9,09	1	2,27	0	0,00	2.006
\web\META-INF\con...	2	2	100,00	0	0,00	0	0,00	0	0,00	84
\web\WEB-INF\web....	24	20	83,33	4	16,67	0	0,00	0	0,00	841
\web\WEB-INF\tiles....	6	5	83,33	1	16,67	0	0,00	0	0,00	321
TOTAL COUNT FOR...	1.442	1.252	86,82	36	2,50	154	10,68	0	0,00	78.499

Figura III.20: Líneas de código .xml en Struts 2



File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\web\index.jsp	39	28	71,79	11	28,21	0	0,00	0	0,00	656
\web\JS\Untitled-3.asp	27	20	74,07	7	25,93	0	0,00	0	0,00	992
\web\JS\Untitled-1.h...	47	32	68,09	6	12,77	9	19,15	0	0,00	1.089
\web\PgHtml\Menu...	37	37	100,00	0	0,00	0	0,00	0	0,00	1.724
\web\PgHtml\menu....	52	51	98,08	1	1,92	0	0,00	0	0,00	2.527
\web\WEB-INF\jsp\...	72	64	88,89	3	4,17	5	6,94	0	0,00	3.313
\web\WEB-INF\jsp\...	72	64	88,89	3	4,17	5	6,94	0	0,00	3.329
\web\WEB-INF\jsp\...	72	64	88,89	3	4,17	5	6,94	0	0,00	3.312
\web\WEB-INF\jsp\...	47	46	97,87	1	2,13	0	0,00	0	0,00	1.571
\web\WEB-INF\jsp\...	47	46	97,87	1	2,13	0	0,00	0	0,00	1.577
\web\WEB-INF\jsp\...	48	47	97,92	1	2,08	0	0,00	0	0,00	1.559
TOTAL COUNT FOR...	1.120	998	89,11	74	6,61	48	4,29	0	0,00	43.298

Figura III.21: Líneas de código .html y .jsp en Struts 2

TABLA III.XXXIV

Comparación del indicador Número de líneas de código

FRAMEWORK	NUMERO DE LINEAS DE CODIGO			TOTAL
	.java	.xml	.html y .jsp	
Struts 2	410	1442	1120	**2972**
Spring MVC	102	1480	1064	**2646**

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.3.1.2. Resultados

TABLA III.XXXV

Resultados de la variable Codificación

Frameworks / Indicadores	Struts 2	Spring MVC
Facilidad para entender el código	2	3

Número de líneas de código	3	4
Total	**5**	**7**

Elaborado por: Ximena Moposita, Mercedes Morán

$$C_{Struts2} = \sum_{i=1}^{2} VPEi = 2 + 3 = 5$$

$$C_{SpringMVC} = \sum_{i=1}^{2} VPEi = 3 + 4 = 7$$

3.4.2.3.1.3. Interpretación

La facilidad en la codificación es importante para tener un mejor entendimiento cuando otra persona tenga que asumir un proyecto web de otro desarrollador. Ambos frameworks hacen una división entre controladores, modelos de JavaBeans y vistas, haciendo una interpretación más clara.

El número de líneas de código del framework Spring MVC (2646) es menor al de Struts2 (2972) lo que hace que esta diferencia significativa porque si el código es pequeño el desarrollo es más rápido y con menos errores permitiendo que el código sea más fácil de entender mantener y mejorar.

3.4.2.3.2. Tiempo de aprendizaje

3.4.2.3.2.1. Análisis de los indicadores de tiempo de aprendizaje

3.4.2.3.2.1.1. Comprensibilidad

Este indicador se refiere al tiempo para comprender el funcionamiento y los componentes de cada uno de los frameworks.

TABLA III.XXXVI

Comparación del indicador Comprensibilidad

FRAMEWORK	COMPRENSIBILIDAD
Struts 2	El tiempo requerido fue de 2 días, del 05/03/2012 al 06/03/2012
Spring MVC	El tiempo requerido fue de 3 días, del 02/04/2012 al 04/04/2012

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.3.2.1.2. Facilidad de aprendizaje

Este indicador determina el tiempo necesario para obtener un nivel de conocimiento suficiente para desarrollar el modulo de prueba usando el framework en base a la experiencia personal de las autoras.

TABLA III.XXXVII

Comparación del indicador Facilidad de Aprendizaje

FRAMEWORK	FACILIDAD DE APRENDIZAJE
Struts 2	El tiempo de aprendizaje de este framework requiere un esfuerzo moderado para alcanzar la experiencia necesaria para el desarrollo del modulo de prueba. El tiempo fue de 43 días del 07/03/12 al 04/05/2012.
Spring MVC	El desarrollo de este modulo requirió esfuerzo moderado en el aprendizaje para conseguir los resultados deseados. El tiempo fue de 44 días del 05/04/2012 al 05/06/2012.

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.3.2.2. Resultados

TABLA III.XXXVIII

Resultados de la variable Tiempo de Aprendizaje

Frameworks / Indicadores	Struts 2	Spring MVC
Facilidad de aprendizaje	3	3
Comprensión	3	3
Total	**6**	**6**

Elaborado por: Ximena Moposita, Mercedes Morán

$$C_{Struts2} = \sum_{i=1}^{2} VPEi = 3 + 3 = 6$$

$$C_{SpringMVC} = \sum_{i=1}^{2} VPEi = 2 + 3 = 5$$

3.4.2.3.2.3. Interpretación

La experiencia durante el desarrollo del modulo de prueba de los dos frameworks requirió tener conocimientos previos de programación en Java y estar familiarizado con el uso de las principales tecnologías JavaEE. Al iniciar el estudio los dos frameworks presentaron cierta dificultad, Struts2 fue un poco menos complicado de aprender que Spring MVC pero una vez comprendido resulta ligeramente más rápido de desarrollar.

3.4.2.3.3. Tiempo

3.4.2.3.3.1. Análisis de los indicadores de tiempo de desarrollo

Para el análisis de esta variable se hará uso del diagrama Gant para contabilizar los días utilizados durante el desarrollo del modulo de prueba.

3.4.2.3.3.1.1. Diagrama Gant de Struts2 con los indicadores

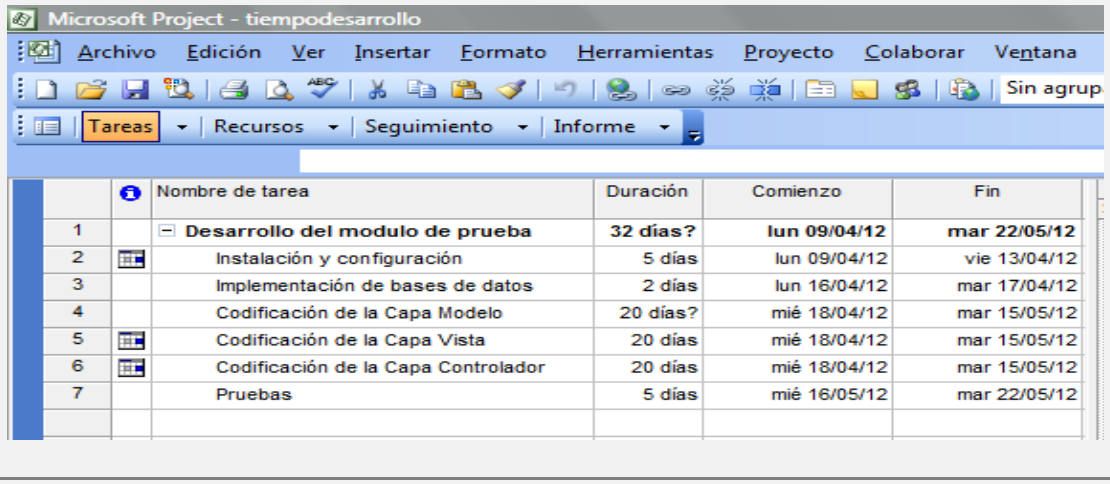


Figura III.22: Diagrama Gant de Struts 2

3.4.2.3.3.1.2. Diagrama Gant de Spring MVC

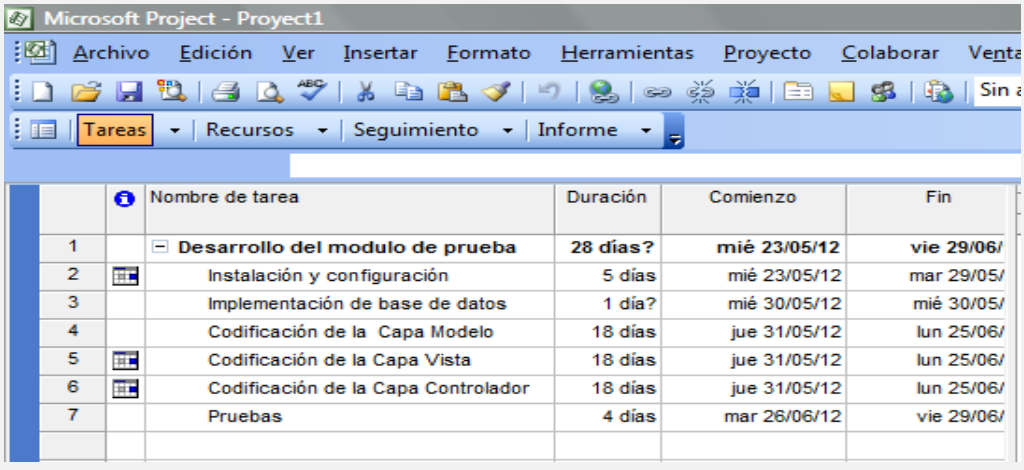


Figura III.23: Diagrama Gant de Spring MVC

3.4.2.3.3.2. Resultados

TABLA III.XXXIX

Análisis comparativo de la variable Tiempo de desarrollo

TIEMPO	Struts 2	Spring MVC
Fecha inicio:	09/04/2012	23/05/2012
Fecha fin:	22/05/2012	29/06/2012
Cantidad de días:	32	28

Elaborado por: Ximena Moposita, Mercedes Morán

TABLA III.XXXX

Resultados de la variable Tiempo de desarrollo

Frameworks / **Indicadores**	Struts 2	Spring MVC
Instalación y configuración Implementación de base de datos Capa modelo Capa vista Capa controlador Pruebas	3	4
Total	**3**	**4**

Elaborado por: Ximena Moposita, Mercedes Morán

3.4.2.3.3.3. Interpretación

El número de días que tomó para desarrollar el modulo de prueba de Spring MVC es menor al de Struts 2 reflejando que nivel de complejidad del framework es ligeramente menor que el otro.

3.4.3. Resultados del análisis de los frameworks Struts 2 y Spring MVC

TABLA III.XXXXI

Resultado del análisis de los frameworks Struts 2 y Spring MVC

Frameworks / **Parámetros**	Struts 2	Spring MVC
Patrón MVC	42	44
Otras características	7	8
Tiempo de desarrollo	14	17
Total	**63**	**69**

Elaborado por: Ximena Moposita, Mercedes Morán

TABLA III.XXXXII

Resultado final del análisis de los frameworks Struts 2 y Spring MVC

Parámetros	**Importancia del parámetro**	**Porcentaje de Struts2**	**Porcentaje de SpringMVC**
Patrón MVC	30%	26,25%	27,5%
Otras características	20%	17,5%	20,0%
Tiempo de desarrollo	50%	35,0%	42,5%

Total	**100%**	**78,75%**	**90%**

Elaborado por: Ximena Moposita, Mercedes Morán

$$\mathbf{VRP1_{Struts2}} = \frac{(\mathbf{VP1_{Struts2}} * P1\%)}{T_{P1}} = \frac{(42 * 30\%)}{48} = 26{,}25\%$$

$$\mathbf{VRP1_{SpringMVC}} = \frac{(VP1_{SpringMVC} * P1\%)}{T_{P1}} = \frac{(44 * 30\%)}{48} = 27{,}5\%$$

$$\mathbf{VRP2_{Struts2}} = \frac{(VP2_{Struts2} * P2\%)}{T_{P2}} = \frac{(7 * 20\%)}{8} = 17{,}5\%$$

$$\mathbf{VRP2_{SpringMVC}} = \frac{(VP2_{SpringMVC} * P2\%)}{T_{P2}} = \frac{(8 * 20\%)}{8} = 20\%$$

$$\mathbf{VRP3_{Struts2}} = \frac{(VP3_{Struts2} * P3\%)}{T_{P3}} = \frac{(14 * 50\%)}{20} = 35\%$$

$$\mathbf{VRP3_{SpringMVC}} = \frac{(VP3_{SpringMVC} * P2\%)}{T_{P3}} = \frac{(17 * 50\%)}{20} = 42\%$$

Donde:

P1 = Parámetro 1 = Patrón MVC

P2 = Parámetro 2 = Otras Características

P3 = Parámetro 3 = Tiempo de Desarrollo

P1% = Porcentaje del parámetro 1 = 30%

P2% = Porcentaje del parámetro 2 = 20%

P3% = Porcentaje del parámetro 3 = 50%

$\mathbf{T_{P1}}$ = Total del parámetro 1 = 48

$\mathbf{T_{P2}}$ = Total del parámetro 2 = 8

$\mathbf{T_{P3}}$ = Total del parámetro 3 = 20

$\mathbf{VP1_{Struts2}}$ = Valor del parámetro 1 obtenido del análisis de Struts2 = 42

$\mathbf{VP1_{SpringMVC}}$ = Valor del parámetro 1 obtenido del análisis de SpringMVC = 44

$\mathbf{VP2_{Struts2}}$ = Valor del parámetro 2 obtenido del análisis de Struts2 = 7

$\mathbf{VP2_{SpringMVC}}$ = Valor del parámetro 2 obtenido del análisis de SpringMVC = 8

$\mathbf{VP3_{Struts2}}$ = Valor del parámetro 3 obtenido del análisis de Struts2 = 14

$\mathbf{VP3_{SpringMVC}}$ = Valor del parámetro 3 obtenido del análisis de SpringMVC = 17

$\mathbf{VRP1_{Struts2}}$ = Valor resultante del parámetro 1 obtenido del análisis de Struts2

$\mathbf{VRP1_{SpringMVC}}$ = Valor resultante del parámetro 1 obtenido del análisis de SpringMVC

$\mathbf{VRP2_{Struts2}}$ = Valor resultante del parámetro 2 obtenido del análisis de Struts2

$\mathbf{VRP2_{SpringMVC}}$ = Valor resultante del parámetro 2 obtenido del análisis de SpringMVC

$\mathbf{VRP3_{Struts2}}$ = Valor resultante del parámetro 3 obtenido del análisis de Struts2

$\mathbf{VRP3_{SpringMVC}}$ = Valor resultante del parámetro 3 obtenido del análisis de SpringMVC

3.4.3.1. Diagrama general con porcentajes de resultados

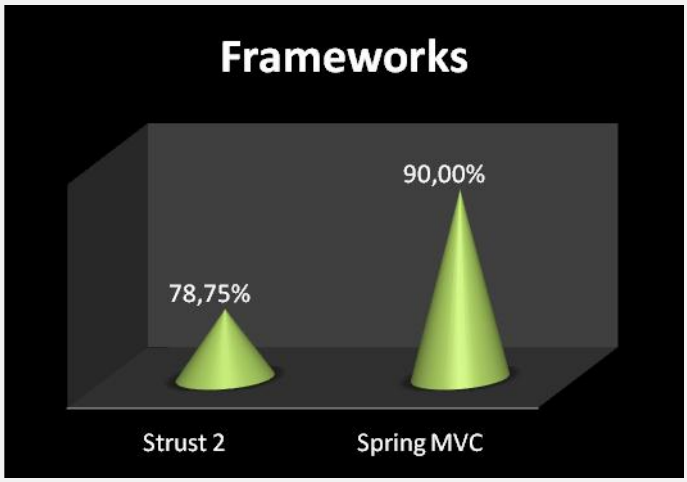


Figura III.24 : Porcentajes de resultados obtenidos

3.4.3.2. Conclusión de los resultados obtenidos.

Según el análisis realizado a cada framework, Spring MVC ha alcanzado un mayor puntaje con un 90% mientras que Struts 2 obtuvo un 78.75%. Puesto que cada framework se cuenta con buenas características en base a los parámetros evaluados se puede concluir que el framework en Java Spring MVC brinda el menor tiempo al momento del desarrollo de aplicaciones Web.

3.5. Comprobación de la Hipótesis.

3.5.1. Análisis de resultados

Se ha realizado el análisis para demostrar la hipótesis que determina que:

"Mediante el análisis entre frameworks de capa de presentación se permitirá seleccionar el más adecuado reduciendo el tiempo de desarrollo de aplicaciones Web."

La misma que se determinó mediante la comparativa de los frameworks seleccionados Spring MVC y Struts2.

3.5.2. Tipo de hipótesis: Causa-Efecto

De acuerdo con la hipótesis planteada se ha podido señalar dos variables:

- ✓ **Variables independientes:** Frameworks de presentación Spring MVC, Struts2.
- ✓ **Variable dependiente**: tiempo de desarrollo.

3.5.3. Operacionalización Conceptual

TABLA III.XXXXIII

Operacionalización Conceptual

Variable	Tipo	Concepto
✓ Framework de presentación Spring MVC	Complejo cualitativo	**Spring MVC** es uno de los módulos del Framework de Spring, y como su propio nombre indica implementa una arquitectura Modelo – Vista – Controlador que se utiliza para

✓ Framework de presentación Struts 2		desarrollar una aplicación web. **Struts2** es un framework web que, implementa el patrón de diseño MVC que nació de la necesidad de evolucionar el código de Struts con el objetivo de simplificar todavía más el desarrollo de la capa de presentación de las aplicaciones web.
Tiempo de desarrollo	Complejo cuantitativo	----------

Elaborado por: Ximena Moposita, Mercedes Morán

3.5.4. Operacionalización metodológica

TABLA III.XXXXIV

Operacionalización Metodológica

Variable	Categoría	Indicadores	Técnicas	Fuentes de verificación
Framework de presentación Spring MVC y Strust2	Requerimientos de utilización	✓ Framework ✓ Base de datos ✓ Herramienta de desarrollo	✓ Utilización directa ✓ Estudio de documentos	✓ Internet ✓ Manuales ✓ Foros web

Tiempo de desarrollo		✓ Tiempo	✓ Observación ✓ Comparación de tiempo	✓ Reportes ✓ Comparación de tiempo que se han utilizado en el desarrollo mediante msproject

Elaborado por: Ximena Moposita, Mercedes Morán

3.5.5. Comprobación del tiempo de desarrollo

Una vez realizado el análisis comparativo de los frameworks en el capítulo III se puede resumir el análisis del parámetro tiempo de desarrollo como se observa a continuación:

3.5.5.1. Datos del parámetro Tiempo para Struts2

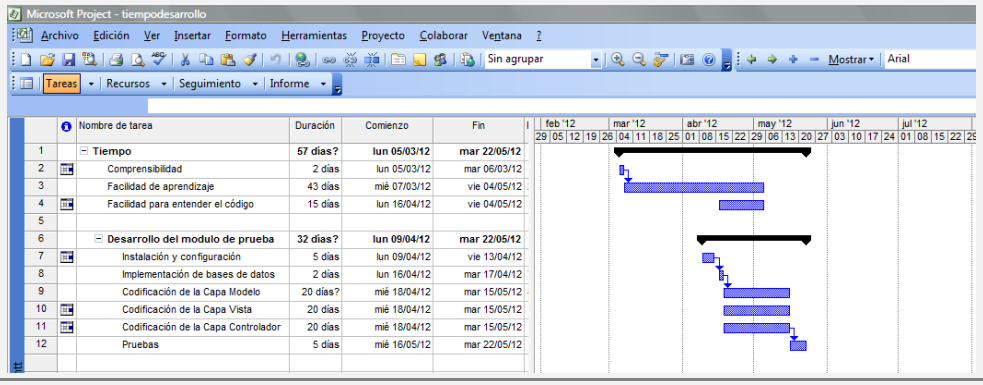


Figura III.25 : Datos del parámetro tiempo para Struts2

3.5.5.2. Datos del parámetro Tiempo para Spring MVC

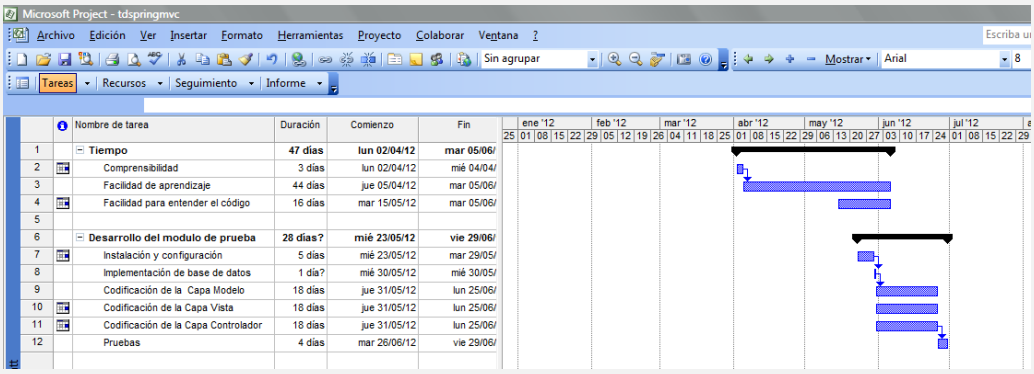


Figura III.26 : Datos del parámetro tiempo para Spring MVC

3.5.5.3. Resumen de datos del parámetro Tiempo entre Strust2 y Spring MVC

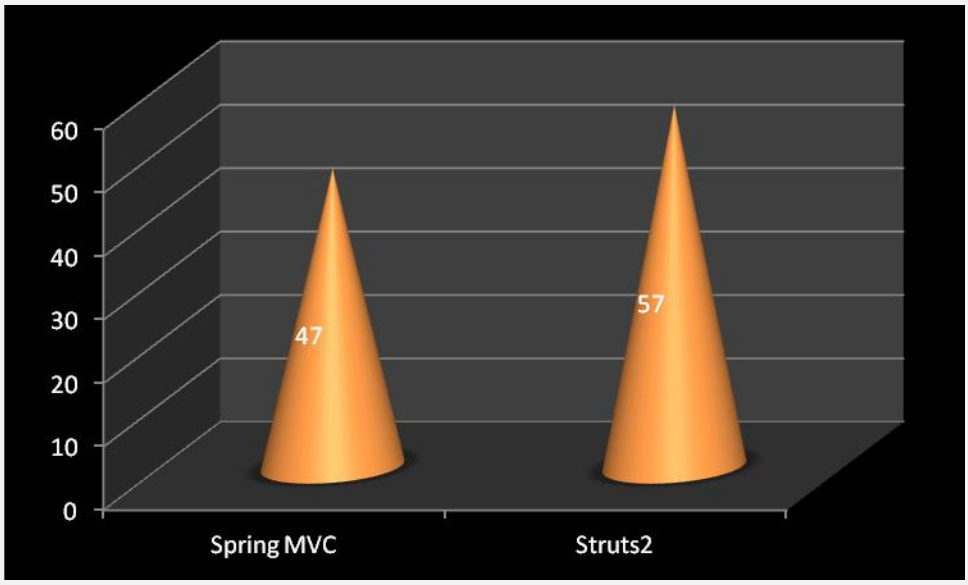


Figura III.27 : Resumen de datos para Struts2 y Spring MVC

3.5.6. Comprobación de la hipótesis

Una vez realizado el estudio previo y posteriormente el análisis comparativo se puede determinar que con el uso del framework de presentación Spring MVC se reduce el tiempo al momento de desarrollar una aplicación web, por lo que se puede confirmar la validez de la hipótesis, es decir, que el resultado es efectivo y se comprueba la hipótesis de la investigación realizada.

CAPÍTULO IV

DESARROLLO DEL SISTEMA DE ARCHIVO GENERAL EN EL GADPCH

4.1. Ingeniería de la información

4.1.1. Definición del ámbito

El Gobierno Autónomo Descentralizado de la Provincia de Chimborazo en la unidad de Archivo General se maneja el control de los documentos que se generan en la institución. Este control lo realiza para llevar un ordenamiento cronológico de los documentos que se almacenan en dicha unidad con el fin de obtener la información necesaria cuando sea solicitada.

Este proceso se lo realiza actualmente mediante matrices que se almacena en Excel donde detallan los ingresos de los documentos al Archivo General provenientes de las coordinaciones, unidades y secciones que conforman el GADPCH, inventario general, tabla de plazos de conservación de documentos y préstamos de documentos.

La información almacena es extensa debido a ciertas redundancias que existe además del consumo excesivo de recursos materiales, talento humano y tiempo.

4.1.2. Identificar Requerimientos

Para solucionar los inconvenientes que se presentan el manejo del sistema de archivo se ha establecido los siguientes requerimientos:

REQ1: El sistema deberá permitir la autenticación de los usuarios.

REQ2: El sistema permitirá el ingreso de un nuevo usuario.

REQ3: El sistema permitirá la modificación de un usuario.

REQ4: El sistema permitirá eliminar un usuario.

REQ5: El sistema permitirá conocer la fecha y hora en la que inicia sesión cada usuario.

REQ6: El sistema permitirá ingresar la información general de documentos (proyectos, oficios, memorandos, reglamentos, etc.) al sistema de las coordinaciones, unidades y secciones.

REQ7: El sistema permitirá ingresar la información específica de documentos al sistema de las coordinaciones, unidades y secciones.

REQ8: El sistema permitirá modificar la información general de documentos al sistema de las coordinaciones, unidades y secciones.

REQ9: El sistema permitirá modificar la información específica de documentos al sistema de las coordinaciones, unidades y secciones.

REQ10: El sistema permitirá verificar la información que llega de los documentos de las coordinaciones, unidades y secciones.

REQ11: El sistema permitirá establecer la ubicación topográfica de los documentos (sección, depósito, estantería, bandeja, caja, carpeta).

REQ12: El sistema permitirá modificar la ubicación topográfica de los documentos (sección, depósito, estantería, bandeja, caja, carpeta).

REQ13: El sistema permitirá ingresar el plazo de conservación de los documentos.

REQ14: El sistema permitirá modificar el plazo de conservación de los documentos.

REQ15: El sistema permitirá realizar préstamos de documentos para funcionarios del GADPCH y personas particulares.

REQ16: El sistema permitirá modificar préstamos de documentos para funcionarios del GADPCH y personas particulares.

REQ17: El sistema permitirá ingresar información para la creación de la guía de formación de archivos.

REQ18: El sistema permitirá modificar información de la guía de formación de archivos.

REQ19: El sistema permitirá generar alertas referentes al límite de tiempo de préstamos de documentos.

REQ20: El sistema permitirá generar alertas referentes a la fecha límite en que se debe conservar un documento.

REQ21: El sistema permitirá obtener un listado de documentos por tipo de documento.

REQ22: El sistema permitirá obtener un listado de documentos por año o coordinación para las coordinaciones, unidades y secciones.

REQ23: El sistema permitirá obtener un listado de documentos con ubicación topográfica para la Unidad de Archivo General.

4.1.3. Estudio de Factibilidad

4.1.3.1. Factibilidad Económica

El GADPCH posee todos los equipos hardware y software necesarios por lo que no se tendrá que realizar ninguna inversión para conseguir recursos**.**

4.1.3.2. Factibilidad Técnica

- ✓ **Recursos humanos**

TABLA IV.XXXXV

Recursos Humanos

Recurso Humano	Cargo
Dr. Julio Santillán	Director
Ing. Danny Velasco	Asesor
Ing. Jaime Zárate	Jefe de la Unidad de Sistemas de GADPCH
Ximena Moposita	Desarrolladora
Mercedes Morán	Desarrolladora

Elaborado por: Ximena Moposita, Mercedes Morán

- ✓ **Recursos Software**

TABLA IV.XXXXVI

Recursos Software

Recurso Software	Característica

Sistema Operativo	Microsoft Windows XP
Servidor de Base de Datos	Mysql
Herramientas de desarrollo	Netbeans 7
Tecnología	Java (Spring MVC)

Elaborado por: Ximena Moposita, Mercedes Morán

4.1.3.3. Factibilidad Operativa

Para desarrollar este sistema se cuenta con el apoyo del jefe de la Unidad de Sistemas Informáticos del GADPCH, de la persona encargada de la Unidad de Sistema de Archivo General y del recurso humano ya mencionado.

4.1.3.4 Factibilidad Legal

Se tiene el permiso de las autoridades respectivas por lo que no existe ningún tipo de impedimento legal para el desarrollo de la aplicación.

4.1.4 Especificación de Requerimientos

Ver Anexo 3

4.2. Análisis del sistema

4.2.1. Casos de Uso del Sistema

Una forma para ver el comportamiento de un sistema desde el punto de vista del usuario es con los casos de usos. Los cuales ayudan a determinar los requerimientos representándolos de manera clara, sencilla y comprensiva las funciones que un sistema puede ejecutar.

Los objetivos de los casos de usos del sistema son:

- ✓ Determinar el comportamiento que el usuario necesita del sistema.
- ✓ Relacionar cada actor con las acciones que quiere conseguir.
- ✓ Aplicar las relaciones que existen entre los casos de uso en el sistema propuesto.

Actores que intervienen y sus funciones

TABLA IV.XXXXVII

Actores y sus funciones

Actor	Función
Usuario Archivo General (administrador del archivo general)	Verifica la información de los documentos que llegan de las coordinaciones. Crea inventario general y tabla de plazos. Ingresa, modifica y elimina información para préstamo de documentos. Consulta de documentos referente de las distintas coordinaciones.
Usuario Coordinaciones (secretaria)	Ingresa, modifica y elimina información de los documentos que se generan en su coordinación. Consulta de documentos referente de las distintas coordinaciones.

Elaborado por: Ximena Moposita, Mercedes Morán

Para la solución del problema se ha identificado los siguientes casos de usos para el sistema de archivo general (SARGE).

- ✓ Autenticación
- ✓ Gestión de información de los documentos
- ✓ Préstamo de documentos
- ✓ Consultas

4.2.2. Detalle de los casos de uso identificados

4.2.2.1. Funcionalidad de los casos de uso

TABLA IV.XXXXVIII

Caso de uso de Autenticación

Identificación:	Autenticación.
Actores:	Usuario (Coordinaciones o Archivo General), Sesión.
Propósito:	Acceder al sistema.
Visión General:	El usuario debe estar registrado en el sistema para que pueda ingresar, modificar o eliminar la información.
Tipo: Primario Esencial	
Curso Típico de Eventos:	
Actores	**Sistema**
1. El usuario ingresa al sistema.	2. Solicita autenticación.

3. Ingresa su nombre y contraseña	4. Verifica los datos ingresados.
Curso Alternativo:	
4. Si los datos ingresados no son correctos solicita que se ingrese nuevamente la información, caso contrario ingresa a su respectiva página de inicio.	

Elaborado por: Ximena Moposita, Mercedes Morán

TABLA IV.XXXXVIX

Caso de uso de Gestión de información de los documentos

Identificación:	Gestión de información de los documentos
Actores:	Usuario Coordinaciones, Usuario Archivo General
Propósito:	Ingreso de información de documentos y manipulación de los mismos.
Visión General:	El usuario Coordinaciones debe recopilar la información de los documentos que se producen e ingresarlo en forma detallada los cuales también puede ser modificados o eliminados, el usuario Archivo General verifica la información en caso de ser correcta crea el Inventario General y Tabla de Plazos.
Tipo: Primario Esencial	
Curso Típico de Eventos:	
Actores	**Sistema**

1. El usuario Coordinaciones se autentica.	2. Presenta las opciones para ingresar la información de los documentos en forma detallada.
3. Decide que acción realizar.	4. Se almacena en la base de datos.
5. El usuario Archivo General se autentica.	6. Presenta las opciones de verificar o ingresar la información de los documentos en forma detallada.
7. Si verifica la información que le llega del usuario Archivo general.	8. Presenta una lista de envíos.
9. Revisa la información si la acepta.	10. Actualiza la base de datos. Y presenta opciones de ingreso.
11. Se ingresa la información aceptada.	12. Se almacena en la base de datos.
Curso Alternativo:	
3. Si los parámetros a ingresar o modificar no son correctos el sistema envía un mensaje de error. 9. Si no acepta la información que le envía un mensaje de rechazo al usuario Coordinaciones. 11. Se ingresa la información al formulario archivo general o tabla de plazos.	

Elaborado por: Ximena Moposita, Mercedes Morán

TABLA IV.XXXXX

Caso de uso de Préstamo de Documentos

<table>
<tr><td>Identificación:</td><td colspan="2">Préstamo de documentos</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario Archivo General</td></tr>
<tr><td>Propósito:</td><td colspan="2">Ingreso de información para préstamo de documentos existentes en la unidad de Archivo General.</td></tr>
<tr><td>Visión General:</td><td colspan="2">El usuario Archivo General ingresa información de una persona que solicita algún documento y del préstamo.</td></tr>
<tr><td colspan="3">Tipo: Primario Esencial</td></tr>
<tr><td colspan="3">Curso Típico de Eventos:</td></tr>
<tr><td colspan="2">Actores</td><td>Sistema</td></tr>
<tr><td colspan="2">1. El usuario Archivo General se autentica.</td><td>2. Presenta las opciones de ingresar información.</td></tr>
<tr><td colspan="2">3. Selecciona la opción préstamo de documentos.</td><td>4. Presenta opciones necesarias para préstamos de documentos</td></tr>
<tr><td colspan="2">5. Ingresa información del solicitante.</td><td>6. Actualiza la base de datos. Y presenta opciones de ingreso.</td></tr>
<tr><td colspan="2">7. Ingresa la información del documento a prestar y del préstamo.</td><td>8. Se almacena en la base de datos.</td></tr>
</table>

Curso Alternativo:

Elaborado por: Ximena Moposita, Mercedes Morán

TABLA IV.XXXXXI

Caso de uso de Consultas

Identificación:	Consultas
Actores:	Usuario (Administrador o Archivo General).
Propósito:	Obtener información de los documentos.
Visión General:	Los usuarios del sistema pueden acceder a obtener información de los documentos.
Tipo: Primario Esencial	
Curso Típico de Eventos:	

Actores	**Sistema**
1. El usuario se autentica.	2. Presenta las opciones de la consulta.
3. Selecciona lo requerido.	4. Presenta reporte de la consulta solicitada.

Curso Alternativo:

1. Si el usuario no está registrado no tendrá acceso a las consultas.

Elaborado por: Ximena Moposita, Mercedes Morán

4.2.2.2. Diagrama de casos de uso

De manera general se presentan los casos de uso que intervienen en funcionamiento del sistema.

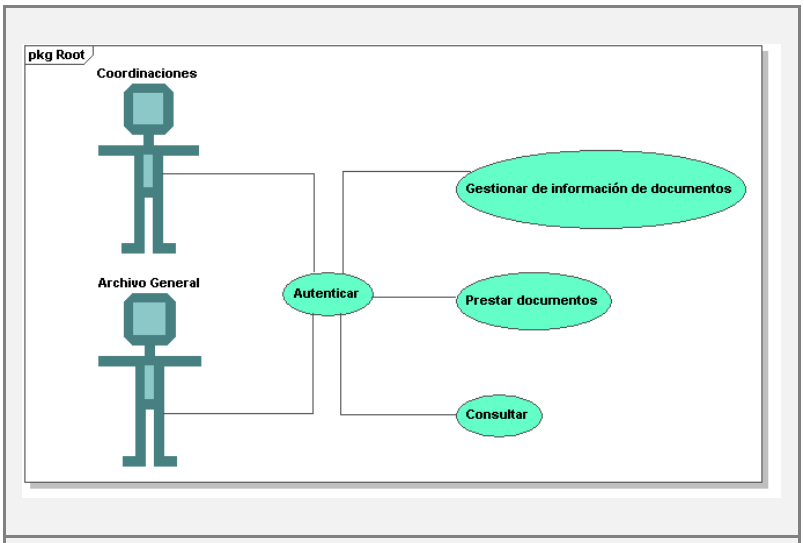


Figura IV.28: Diagrama de casos de uso general

4.2.3. Diagramas de secuencia

El diagrama de secuencias es el núcleo del modelo dinámico, y muestra todos los cursos alternos que pueden tomar todos nuestros casos de uso. Los diagramas de secuencias se componen de 4 elementos que son: el curso de acción, los objetos, los mensajes y los métodos (operaciones).

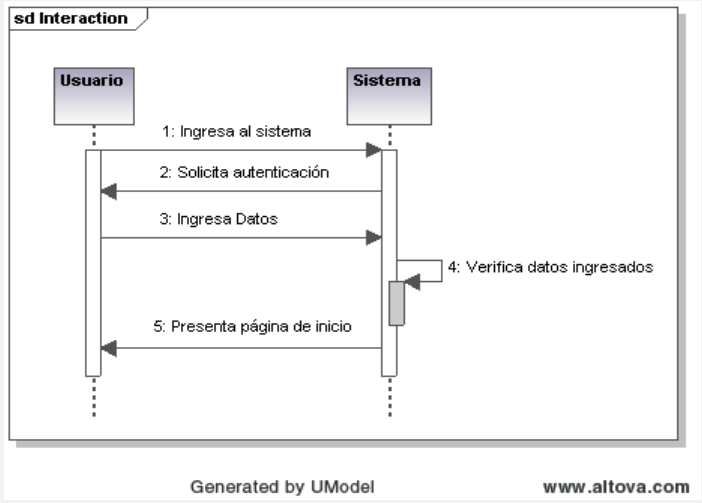


Figura IV.29 : Diagrama de secuencia de autenticación

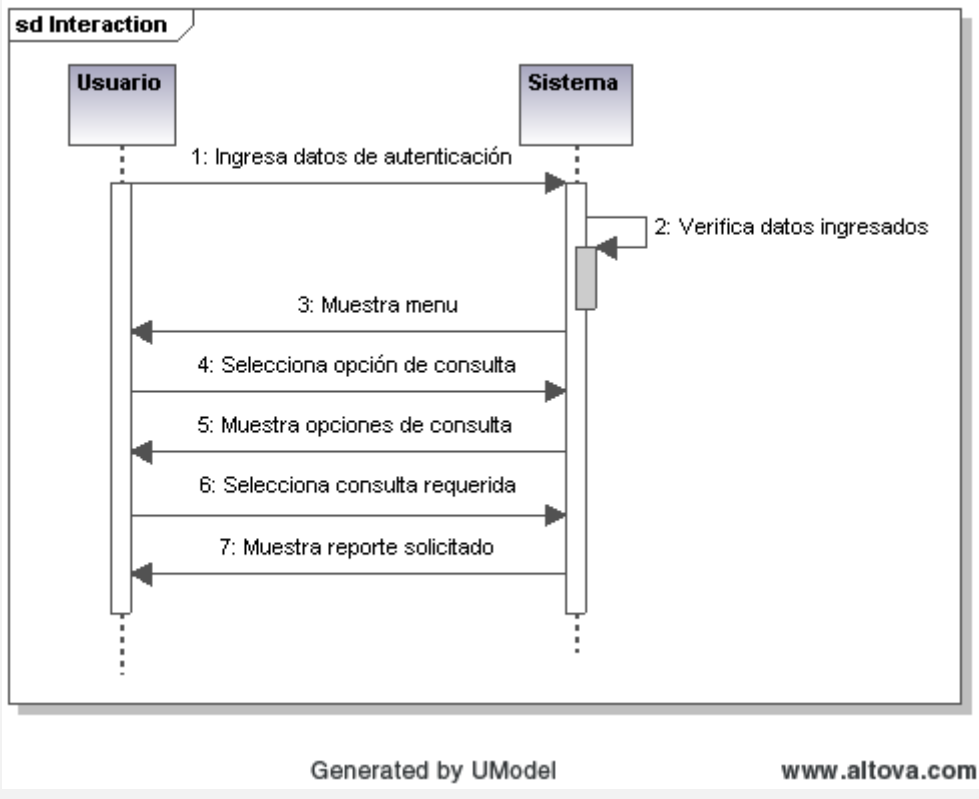


Figura IV.30: Diagrama de secuencia de consultas

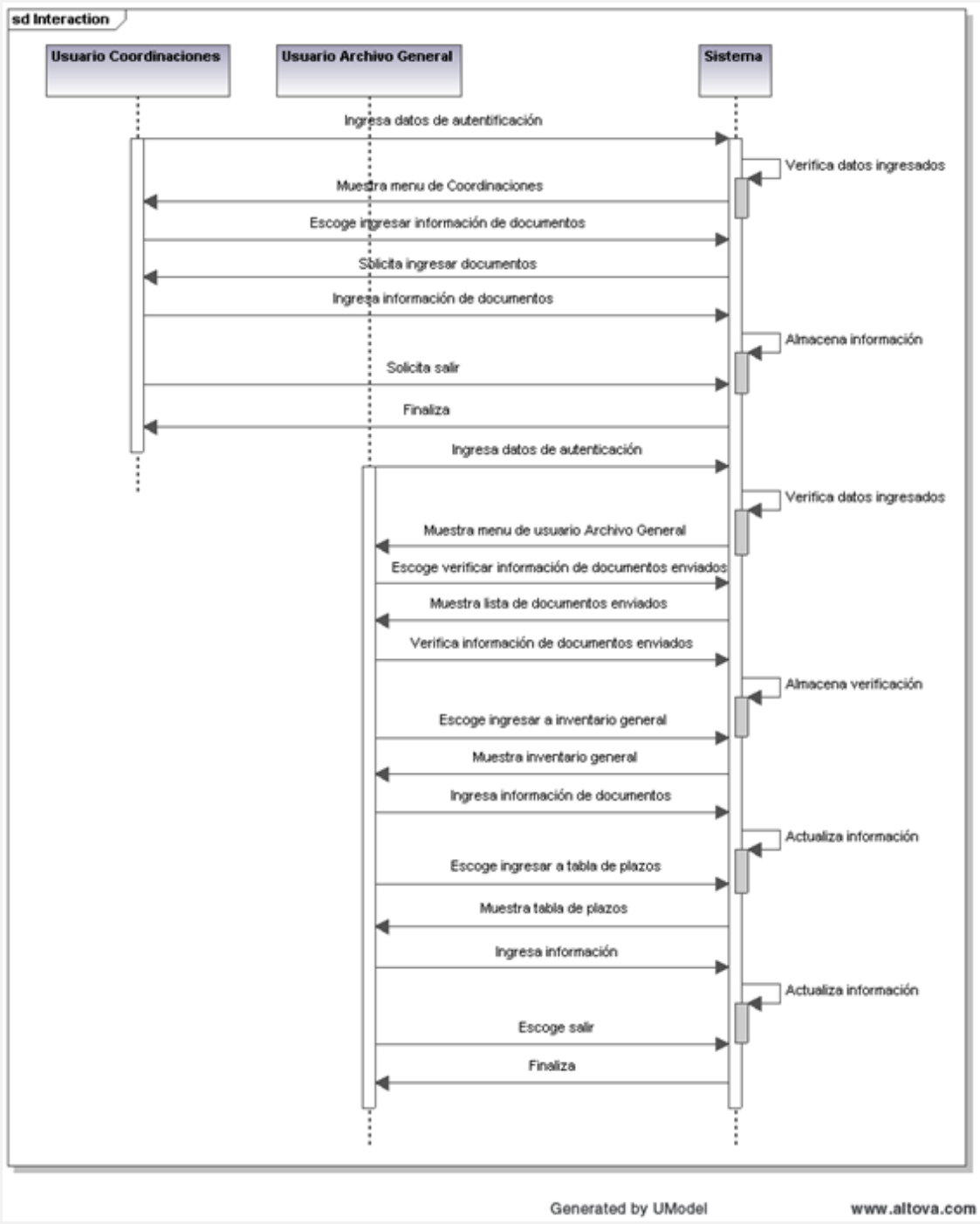


Figura IV.31: Diagrama de secuencia de gestión de información de documentos

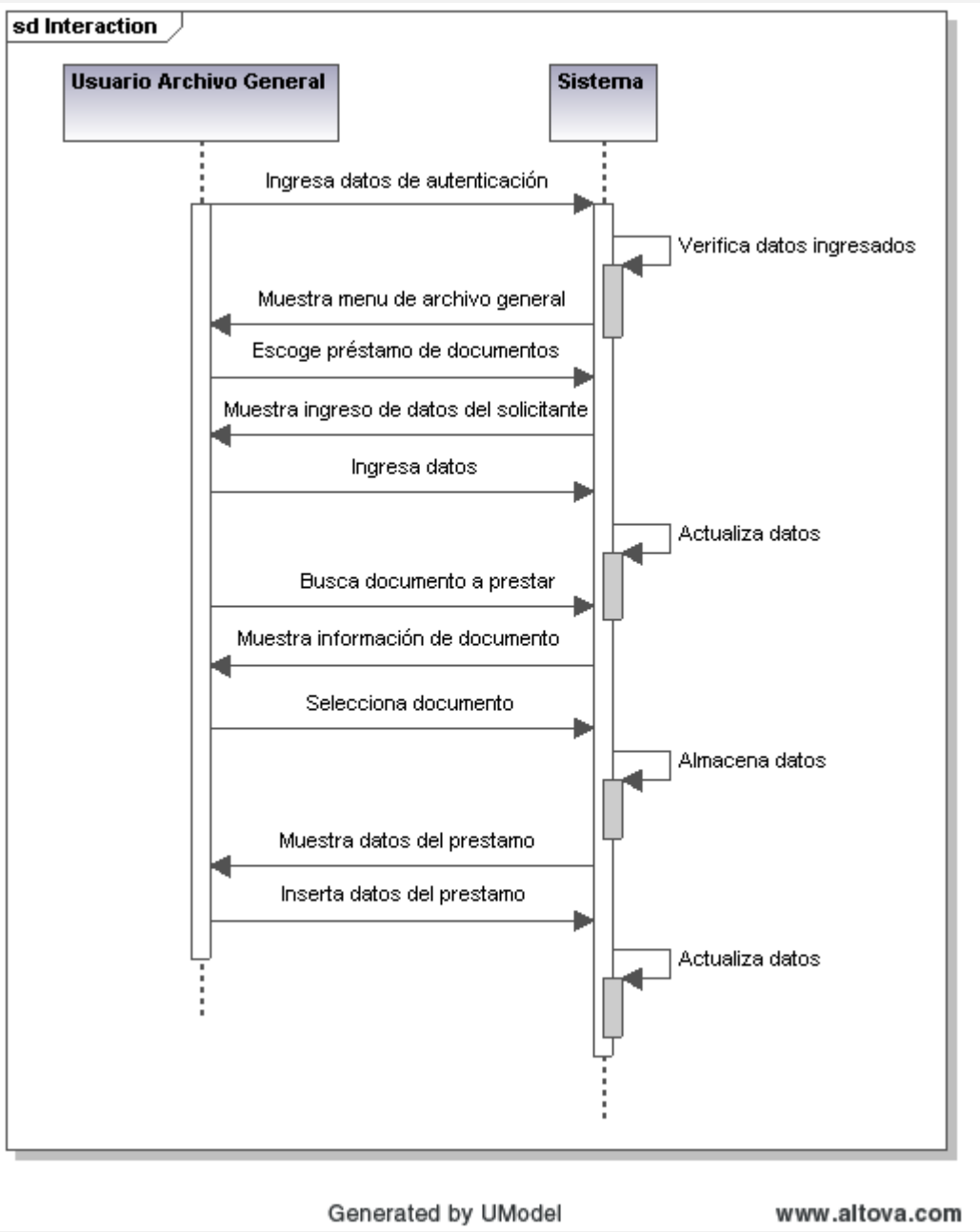


Figura IV.32: Diagrama de secuencia de préstamo de documentos

4.2.4. Diagrama de Colaboración

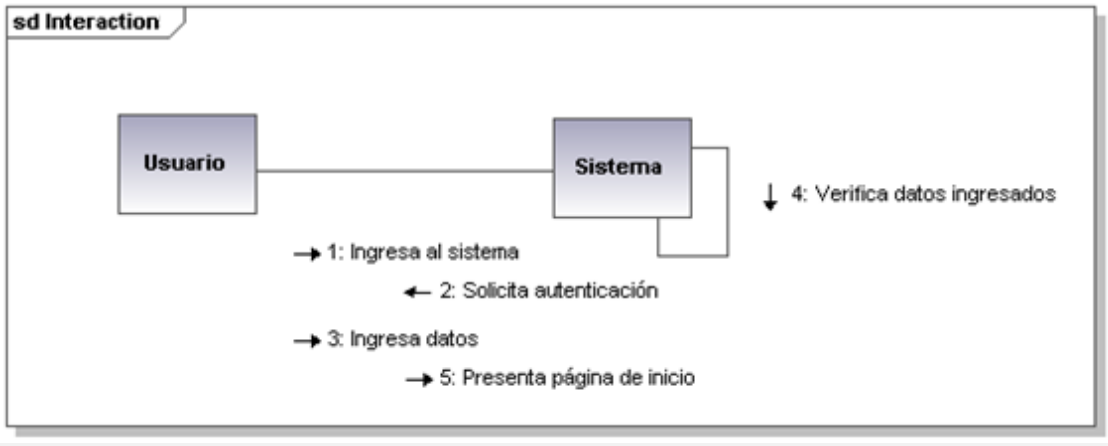


Figura IV.33: Diagrama de colaboración de autenticación

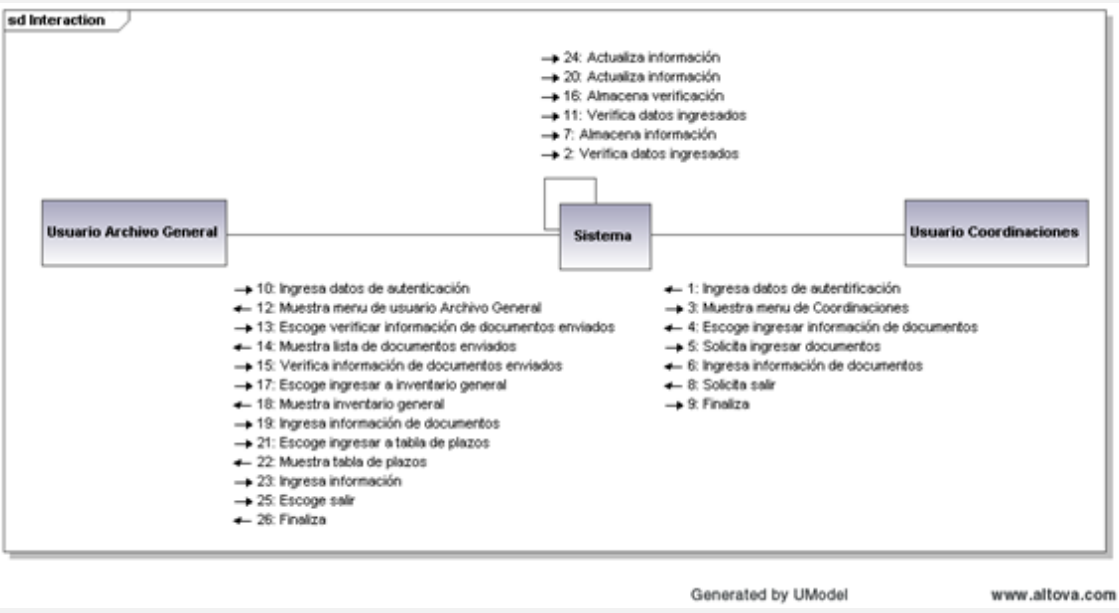


Figura IV.34: Diagrama de colaboración de gestión de información de documentos

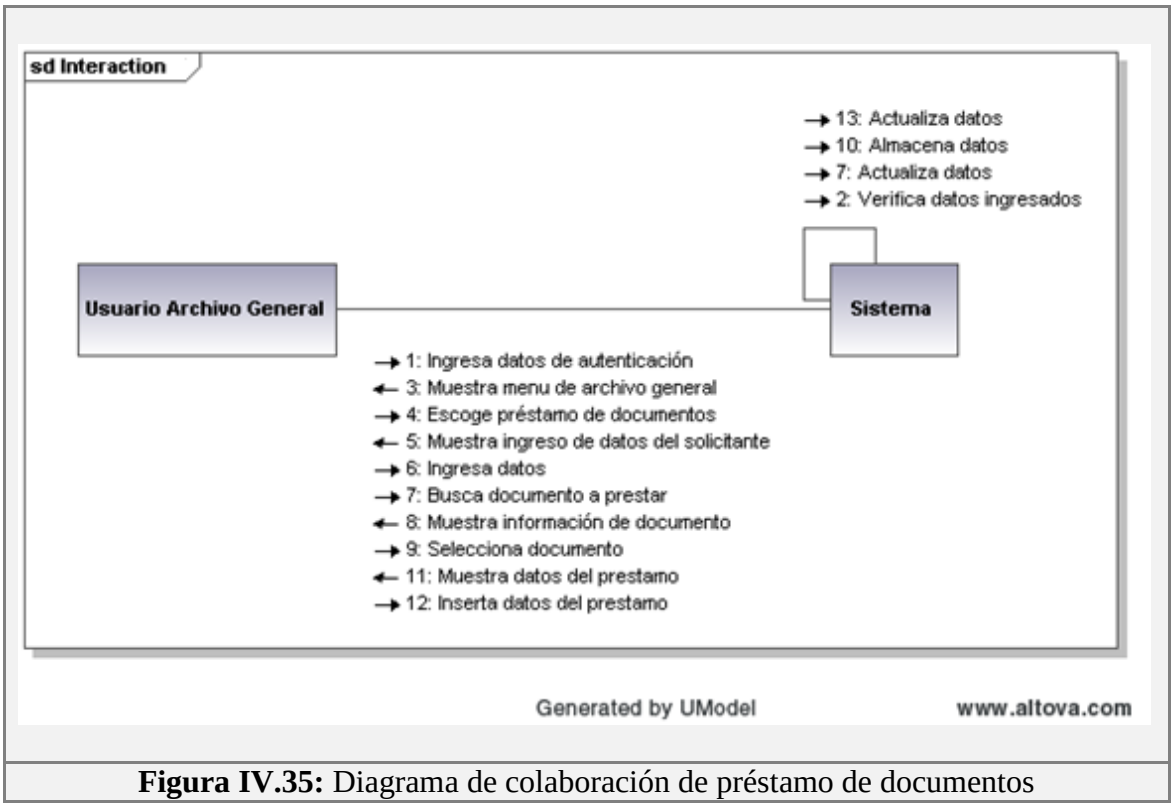


Figura IV.35: Diagrama de colaboración de préstamo de documentos

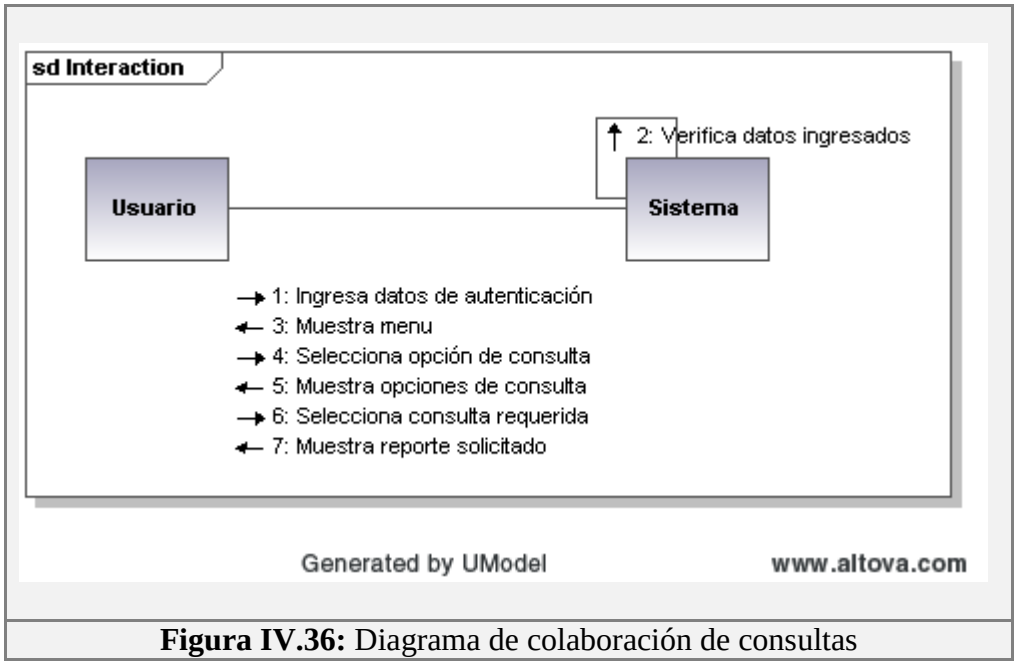


Figura IV.36: Diagrama de colaboración de consultas

4.3. Diseño

4.3.1. Casos de uso reales

TABLA IV.XXXXXII

Caso de uso de Autenticación

Actores	Sistema
1. El usuario ingresa al sistema.	2. Solicita autenticación.
3. Ingresa su nombre y contraseña.	4. Verifica los datos ingresados.

Elaborado por: Ximena Moposita, Mercedes Morán

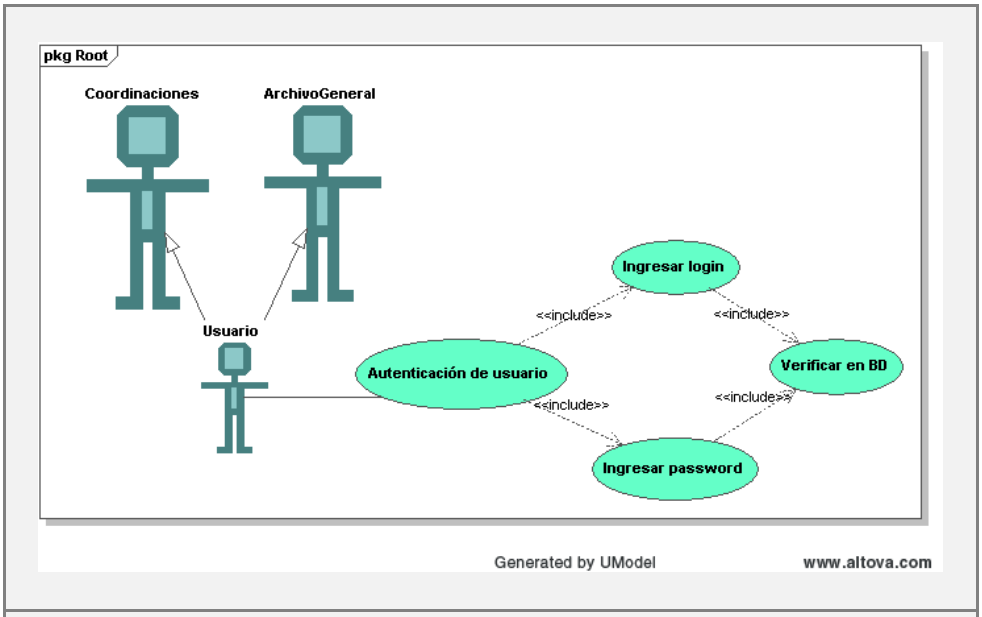


Figura IV.37: Diagrama de casos de uso de autenticación

TABLA XXXXXIII

Caso de uso de Gestión de información de los documentos

Actores	Sistema
1. El usuario Coordinaciones se autentica.	2. Presenta las opciones para ingresar la información de los documentos en forma detallada.
3. Decide que acción realizar.	4. Se almacena en la base de datos.
5. El usuario Archivo General se autentica.	6. Presenta las opciones de verificar o ingresar la información de los documentos en forma detallada.
7. Si verifica la información que le llega del usuario Archivo general.	8. Presenta una lista de envíos.
9. Revisa la información si la acepta.	10. Actualiza la base de datos. Y presenta opciones de ingreso.
11. Se ingresa la información aceptada.	12. Se almacena en la base de datos.

Elaborado por: Ximena Moposita, Mercedes Morán

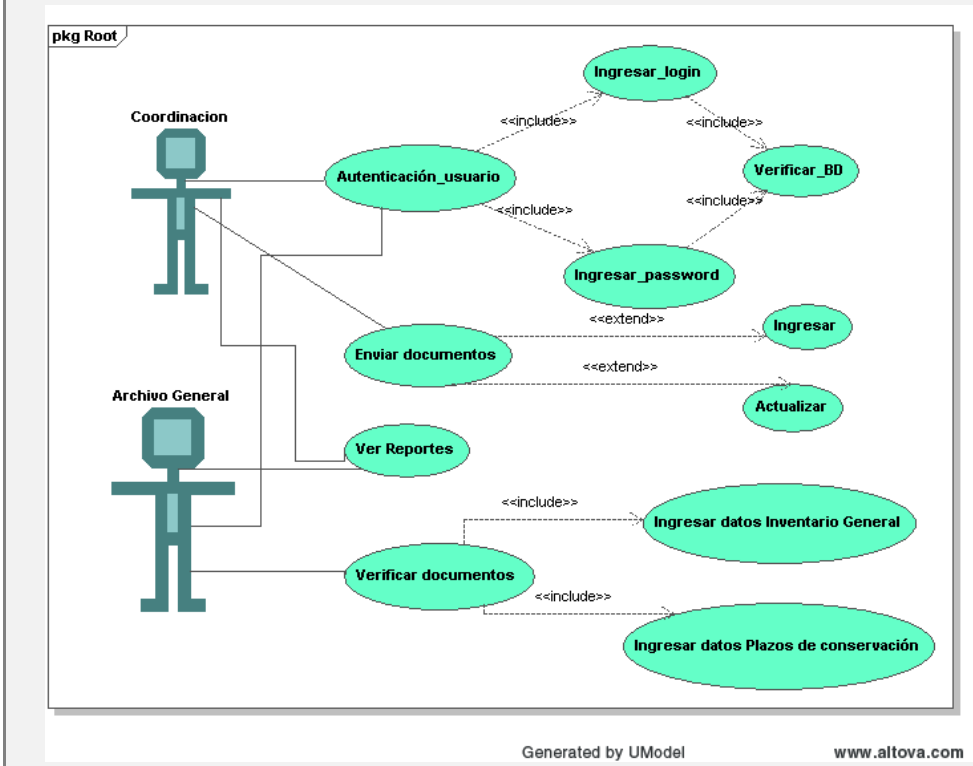


Figura IV.38: Diagrama de casos de uso de gestión de información de documentos

TABLA IV.XXXXXIV

Caso de uso de Préstamo de Documentos

Actores	Sistema
1. El usuario Archivo General se autentica.	2. Presenta las opciones de ingresar información.
3. Selecciona la opción préstamo de documentos.	4. Presenta opciones necesarias para préstamos de documentos

5. Ingresa información del solicitante.	6. Actualiza la base de datos. Y presenta opciones de ingreso.
7. Ingresa la información del documento a prestar y del préstamo.	8. Se almacena en la base de datos.

Elaborado por: Ximena Moposita, Mercedes Morán

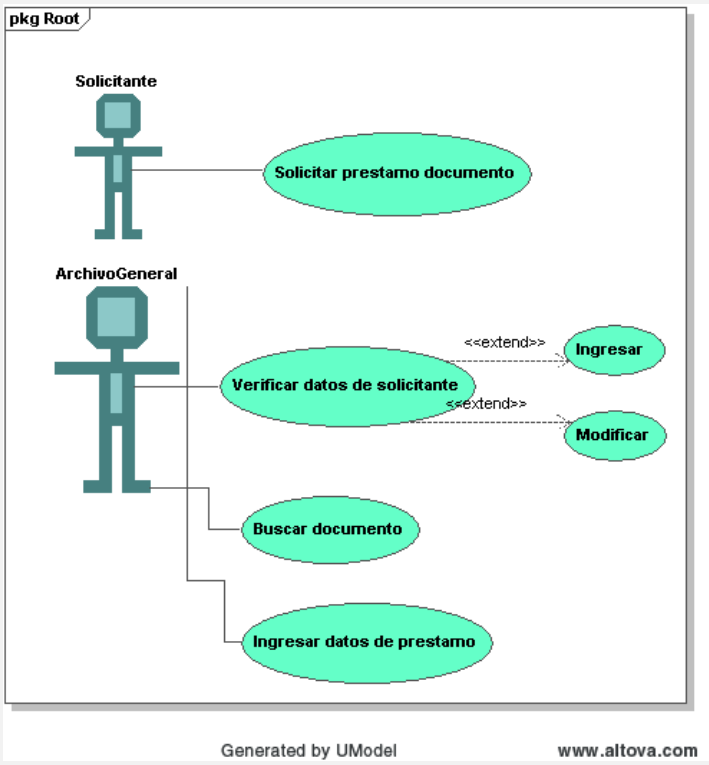


Figura IV.39: Diagrama de casos de uso de préstamo de documentos

TABLA IV.XXXXXV

Caso de uso de Consultas

Actores	Sistema
1. El usuario se autentica.	2. Presenta las opciones de la consulta.
3. Selecciona lo requerido.	4. Presenta reporte de la consulta solicitada.

Elaborado por: Ximena Moposita, Mercedes Morán

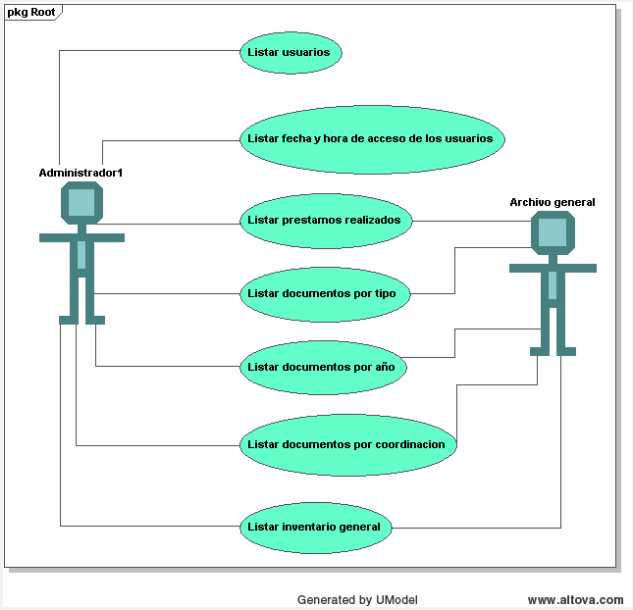


Figura IV.40: Diagrama de casos de uso de consultas

4.3.2. Definición de informes e interfaces de usuario

4.3.2.1. Definición de la información de la interfaz de usuario

Para la interfaz de usuario se ha partido de los casos de usos reales, que ejercen como modelos para definir la interfaz del usuario tanto para las pantallas como para los documentos.

4.3.2.2. Lenguaje de Comunicación

- ✓ **Comunicación con el Usuario:** Es de fácil entendimiento para el usuario debido a que se maneja como una aplicación web.

4.3.3. Diagramas de interacción

4.3.3.1. Diagramas de secuencia

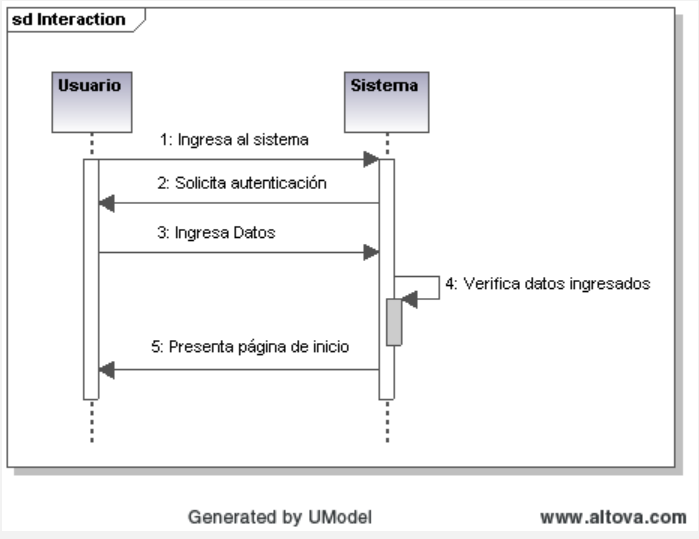


Figura IV.41: Diagrama de secuencia de autenticación

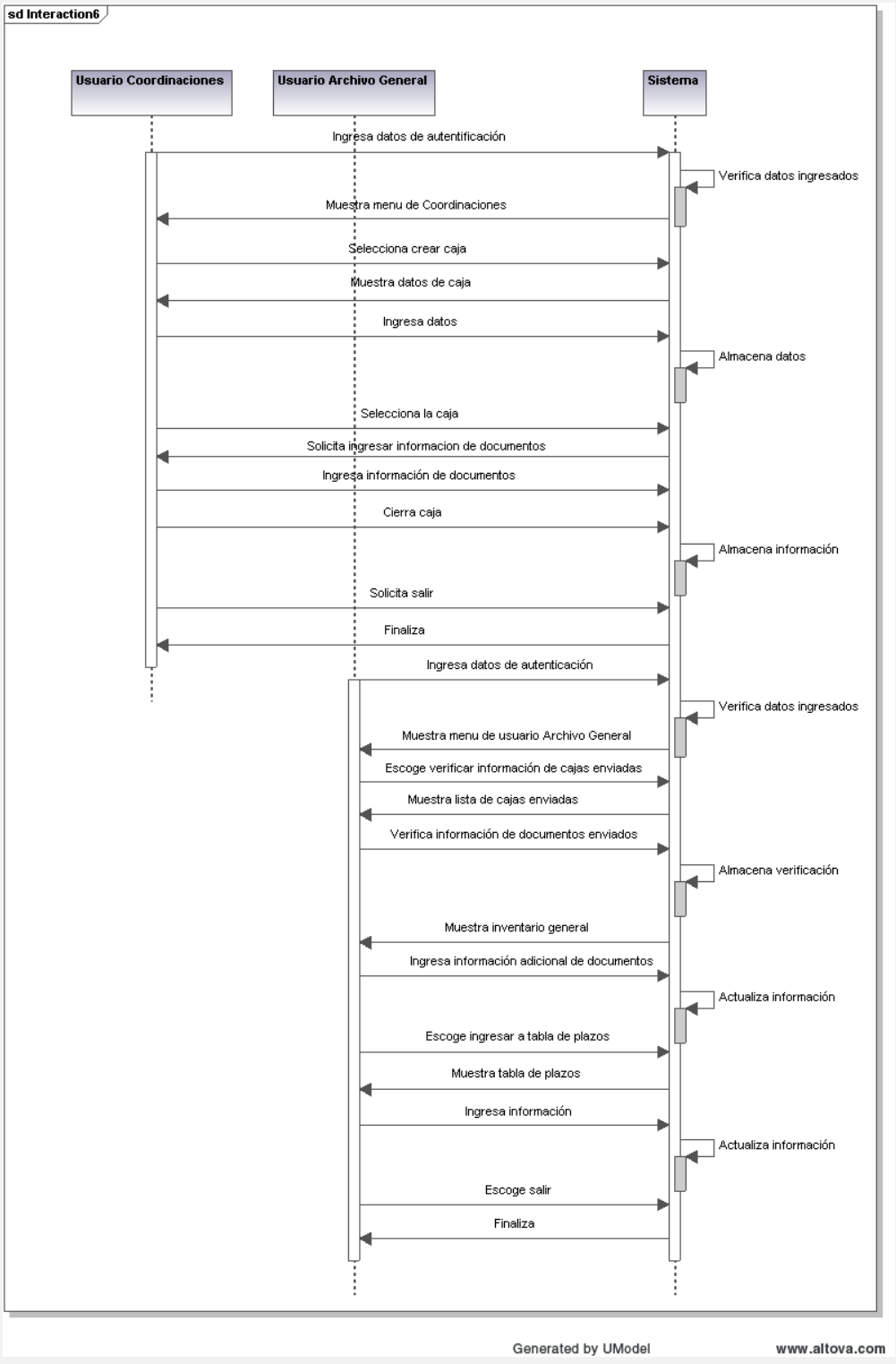


Figura IV.42: Diagrama de secuencia de gestión de información de documentos

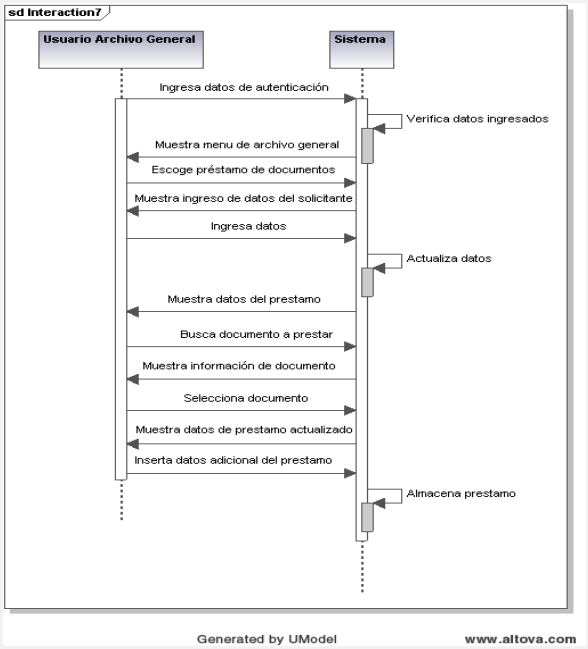


Figura IV.43: Diagrama de secuencia de préstamo de documentos

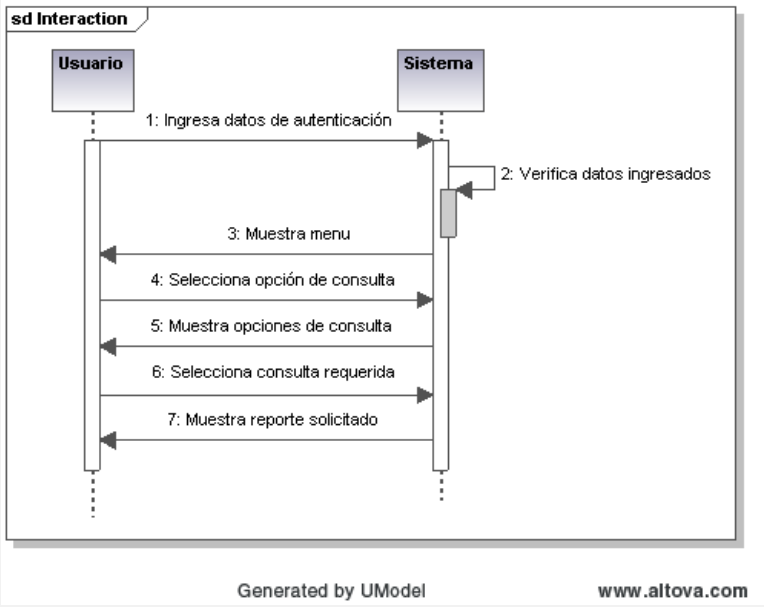


Figura IV.44: Diagrama de secuencia de consultas

4.3.3.2. Diagramas de colaboración

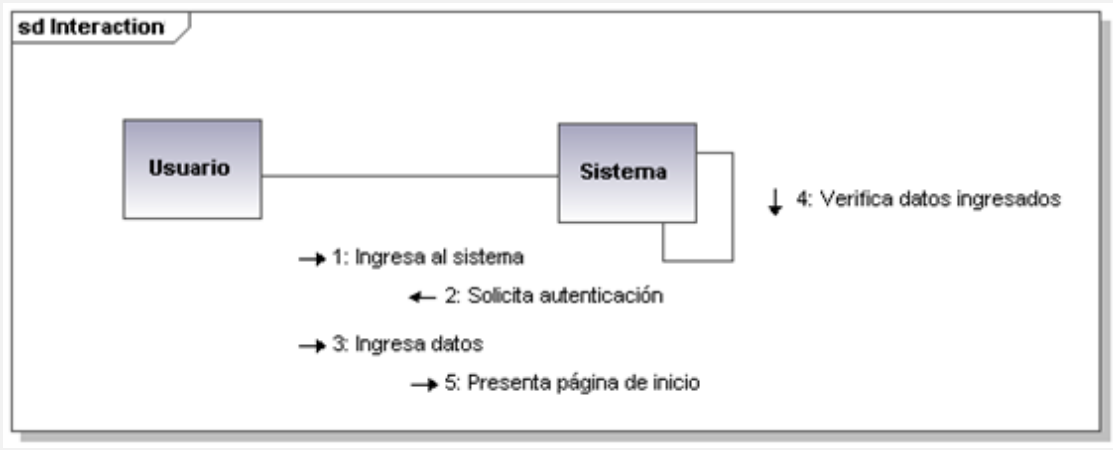


Figura IV.45: Diagrama de colaboración de autenticación

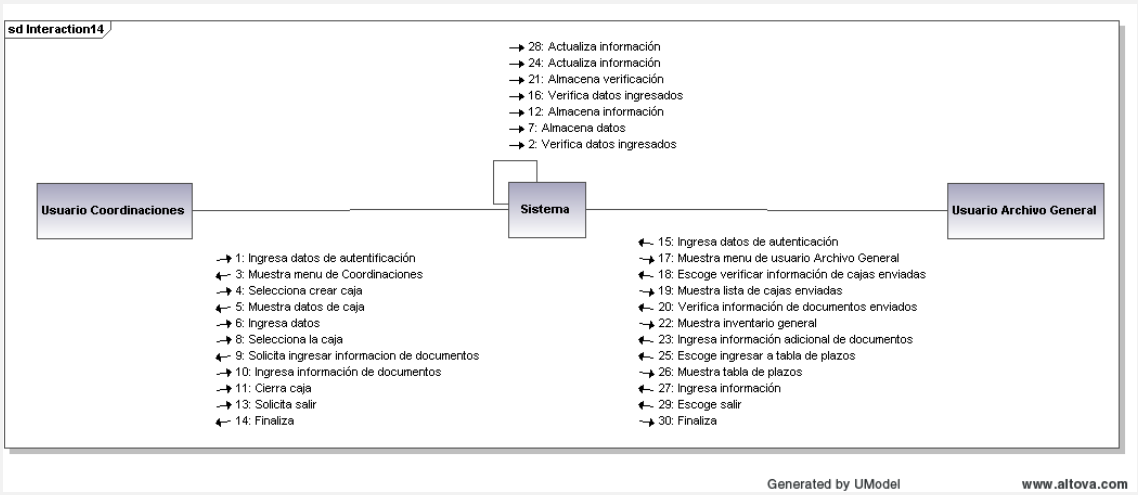


Figura IV.46: Diagrama de colaboración de gestión de información de documentos

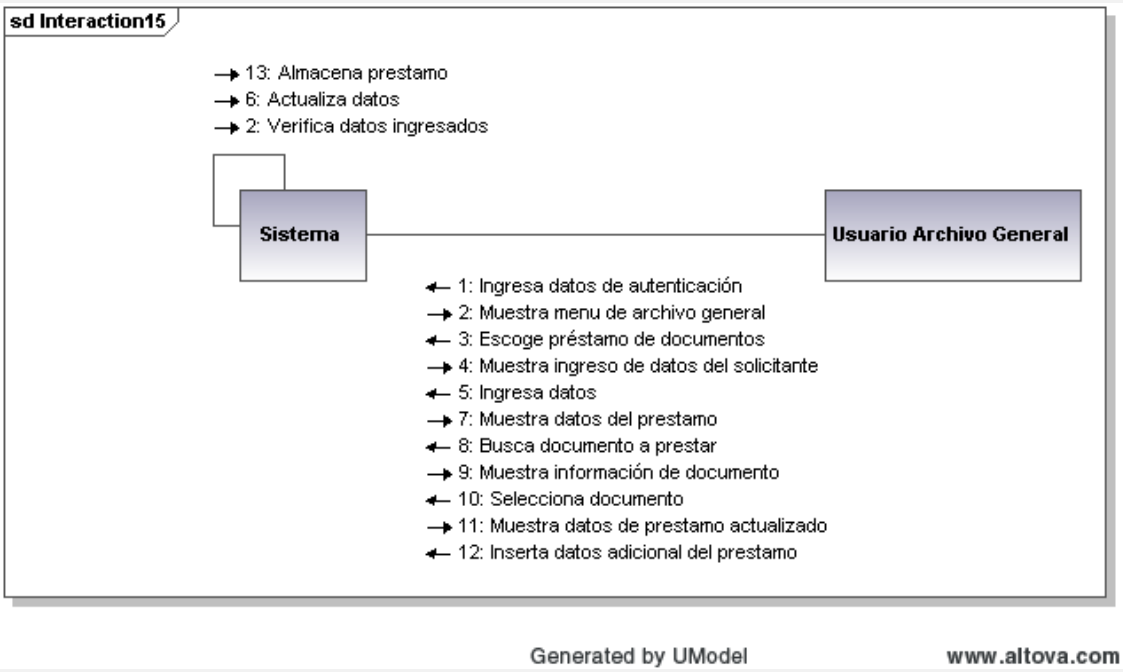


Figura IV.47: Diagrama de colaboración de préstamo de documentos

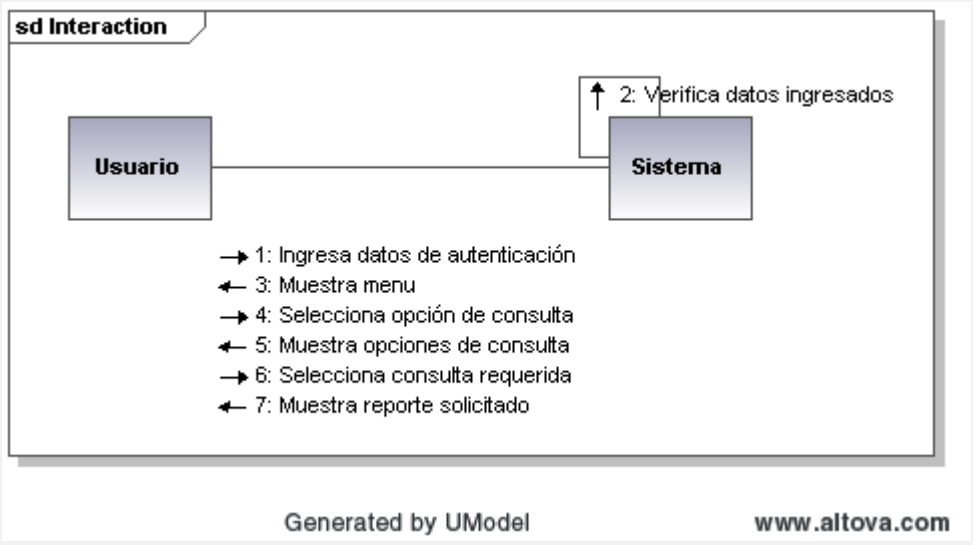


Figura IV.48: Diagrama de colaboración de consultas

4.3.3.3. Diagrama de calles

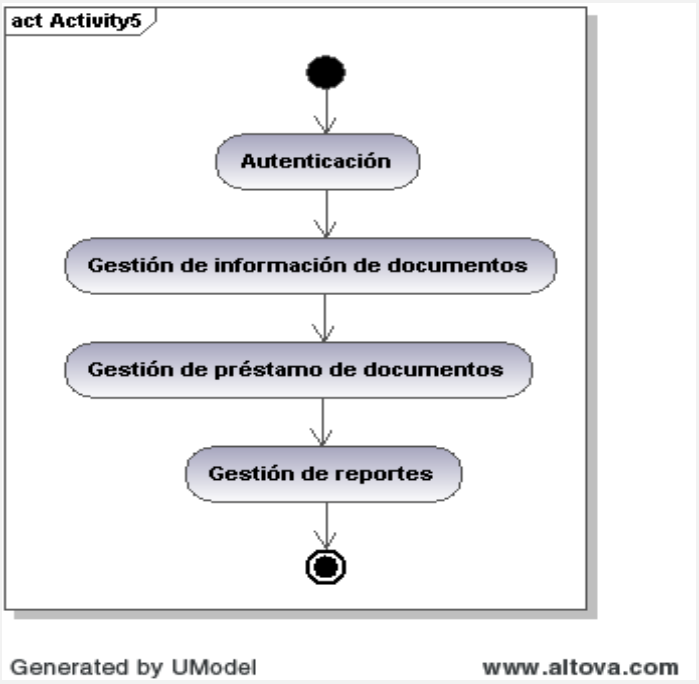


Figura IV.49: Diagrama de calles

4.3.4. Diagrama de clases

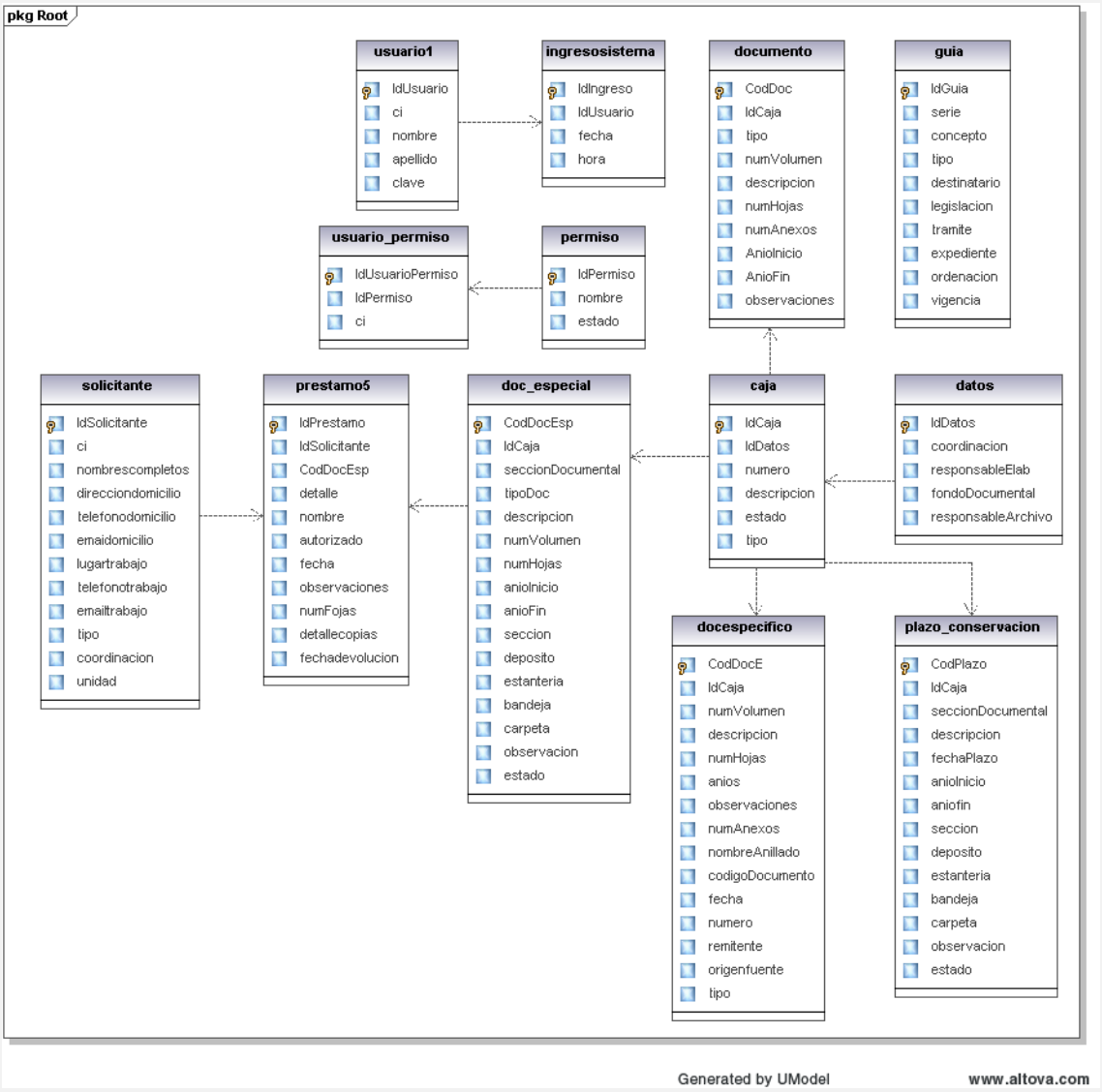


Figura IV.50: Diagrama de clases

4.3.5. Diagrama de Despliegue

4.3.5.1. Diagrama de componentes

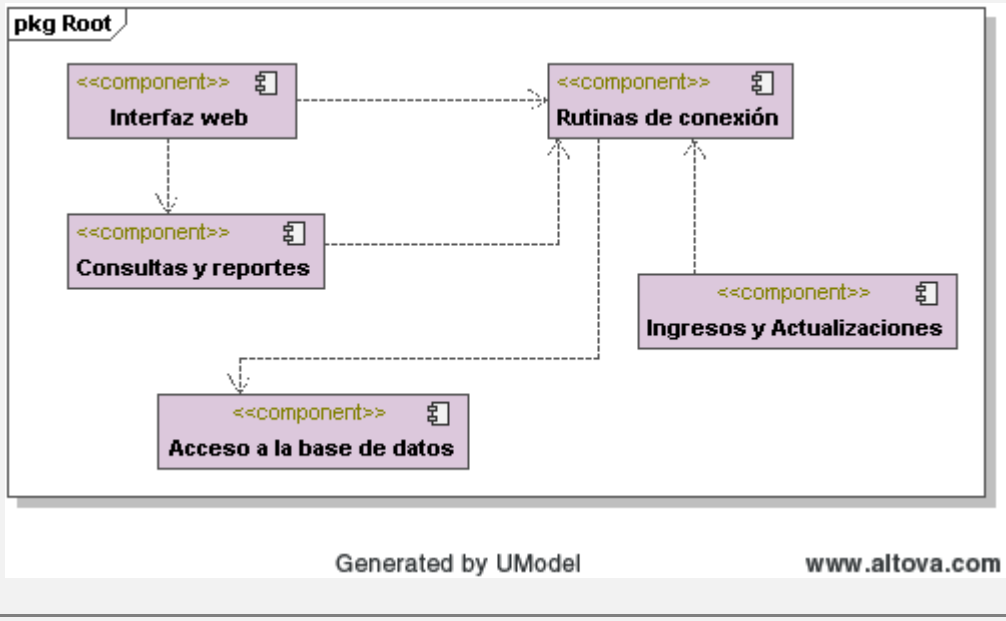


Figura IV.51: Diagrama de componentes

4.3.5.2. Diagrama de nodos

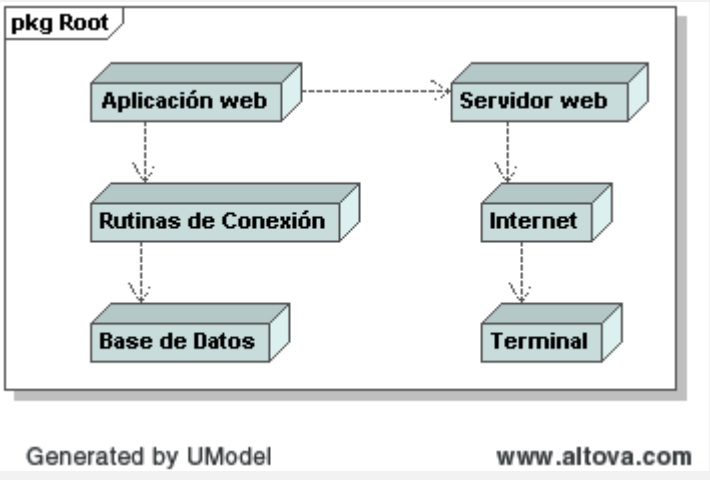


Figura IV.52: Diagrama de nodos

4.4. Implementación y Pruebas

4.4.1. Definición de estándares de programación

Para la realización de la codificación del sistema se ha definido los siguientes estándares:

- ✓ El nombre de las tablas comienzan con letra minúscula, seguido de (_) en caso de que tenga más de una palabra.
- ✓ El nombre de los campos empieza con letra minúscula, seguido de una letra mayúscula en caso de que tenga más de una palabra.
- ✓ El nombre de los atributos de la clase empieza con letra minúscula, seguido de una letra mayúscula en caso de que tenga más de una palabra.
- ✓ Los métodos que interactúan con la base de datos reciben como parámetro de entrada, una lista en el caso de ser necesario.

4.4.2. Pruebas Unitarias

Para asegurar el correcto funcionamiento del sistema se han probado sus clases y sus métodos de forma independiente, enviando datos de entrada desde el código para luego obtenerlos a través de los diferentes métodos para realizar consultas.

4.4.3. Pruebas de modulo y del sistema

Las pruebas finales consistieron en verificar que la información ingresada se vea inmediatamente reflejada en las consultas del sistema, esto sirve para comprobar que la información se está registrando correctamente en la base de datos.

Se provocaron errores intencionales para verificar el correcto funcionamiento del sistema, así como de las funciones de validación de datos, como por ejemplo:

- ✓ Realizar consultas a tablas sin valores
- ✓ Ingresar campos vacíos
- ✓ Buscar información que no existe

CONCLUSIONES

- ✓ Con un conocimiento previo en Java se estableció trabajar con frameworks del mismo pero al existir una gran cantidad de ellos se tomo en cuenta una encuesta que determino que frameworks prefieren usar los desarrolladores de esta manera se descartaron opciones ya que es muy difícil evaluar a todos. Enfatizando que todos los frameworks de esta encuesta usan el patrón MVC siendo los más viables y reconocidos: Spring MVC, Struts2, JSF, Tapestry y Cocoon.
- ✓ Con el estudio realizado a las características básicas, ventajas y desventajas de los framewoks evaluados se pudo tener una idea preliminar de lo que es cada uno de ellos ayudando de esta manera a determinar algunos de los parámetros que se usaron para el análisis comparativo.
- ✓ Al momento de plantear los parámetros de evaluación para elaborar la comparativa de frameworks de presentación se basó esencialmente en la búsqueda del que mejor se adapte a las nuestras necesidades ayudando a cumplir con los objetivos y la comprobación de la hipótesis que en este caso es detectar el más adecuado desarrollando aplicaciones web simplificando tiempo y trabajo pero también que aporte con productividad y eficiencia.
- ✓ Para seguir descartando opciones se realizo un análisis en donde se evaluó la madurez del producto y ligeramente las facilidades al ser aprendidos obteniendo los siguientes porcentajes Spring MVC con un 87.5%, Struts2 91.7%, JSF 95.8%, Tapestry 62.5% y Cocoon con un 54.2%. Mostrando que JSF, Spring MVC y Strusts2 tiene los

porcentajes más altos. Eligiendo en esta fase a Struts2 y Spring MVC a pesar de que JSF tiene un porcentaje muy alto no ha sido tomado en cuenta para la investigación debido a que existen varias tesis en la ESPOCH.

- ✓ En el análisis comparativo final se determino un porcentaje para los frameworks Spring MVC y Struts2 con una calificación de 90% y 78,75% respectivamente en cual Spring MVC se muestra superior a Struts2, determinando que el framework Spring MVC es el más adecuado reduciendo el tiempo de desarrollo de aplicaciones Web.
- ✓ Spring MVC es muy flexible, ya que implementa toda su estructura mediante interfaces, provee interceptores y controllers que permiten interpretar y adaptar el comportamiento común en el manejo de múltiples requests.
- ✓ Se utilizó el framework de presentación Spring MVC en el desarrolló la aplicación web para la unidad de Archivo General del Gobierno Autónomo Descentralizado de la Provincia de Chimborazo, ya que este ofrece mejores beneficios en cuanto a la rapidez de su desarrollo.
- ✓ La aplicación Web SARGE desarrollada, cumple con todos los requerimientos señalados, brindando a los usuarios finales mejor control y organización en la información.

RECOMENDACIONES

- ✓ Se recomienda la utilización del framework Spring MVC al momento de crear aplicaciones web si se desea obtener menor tiempo en su desarrollo debido a las características que este posee.
- ✓ Realizar un análisis previo a la utilización de cualquier framework para que seleccione el que mejor se acople a los requerimientos de su aplicación web.
- ✓ Es importante que un desarrollador esté al día actualizando sus conocimientos en cada mejora que se le añada al framework de su elección.
- ✓ Estudiar características y funciones ofrecidas por los frameworks de presentación de forma que al seleccionar uno de ellos se aproveche al máximo sus cualidades y al mismo tiempo ayudaran al desarrollo rápido de aplicaciones en tiempos cortos.
- ✓ Para iniciar un análisis comparativo se debe seleccionar cuidadosamente los frameworks involucrados en el mismo, porque deben tener características semejantes para compararlas sin ningún inconveniente.
- ✓ Al momento de realizar una comparativa se requiere de una definición clara de cada parámetro que ayude a la comprobación de la hipótesis del estudio realizado.
- ✓ Plantear correctamente con el usuario final todos los requisitos que desea para no perder tiempo en el desarrollo de la aplicación.

RESUMEN

Se investigó el análisis de frameworks de presentación para el desarrollo de Aplicaciones Web en Java, y su aplicación se la realizó en la Unidad de Archivo General del Gobierno Autónomo Descentralizado de la Provincia de Chimborazo. Se eligió cinco frameworks de presentación: Spring MVC, Struts2, JSF, Tapestry y Cocoon mediante una encuesta en una página web dirigida a desarrolladores web. Con dicho análisis previo se busco obtener los dos mejores frameworks que reflejen facilidad y suficientes características para uso efectivo, para luego terminar con un análisis más profundo entre Spring MVC y Struts 2 con los parámetros de manejo del Patrón MVC, otras características y el más importante tiempo de desarrollo que ayudo a la comprobación de la hipótesis. Observándose que Spring MVC obtuvo una calificación del 90% y Struts2 78.75%, lo que termino que Spring MVC es el más adecuado reduciendo el tiempo de desarrollo de aplicaciones Web.

Para el desarrollo de la aplicación web denominada SARGE, se empleo el framework Spring MVC, el IDE Netbeans, iReport y MySql Server. Se recomienda el aprendizaje y utilización del framework Spring MVC ya que son de gran utilidad al momento de reducir tiempo de desarrollo.

ABSTRACT

The framework analysis about the presentation was researched to the developed of the Web Applications in Java, and its application was realized in de General Achieve Unit of the Decentralized Autonomous Government from Chimborazo Province.It was chosen five presentation frameworks like: Spring MVC, Struts2, JSF, Tapestry and Cocoon, using a survey in a web page applying to the web designers. With a previous analysis it a was researched to obtain two best frameworks that reflect facility and enough characteristics by the effective used of them, finishing with detailed analysis between Spring MVC and Struts2 with the management parameters of MVC pattern, other characteristics and of the most important time of development that helped of verify the hypothesis. Noticing that Spring MVC obtained a rank of 90% and Struts2 78.72% and the conclusion Spring MVC is the most useful reducing the time of the web applications development.

For the development of the application named SARGE, it was used the framework Spring MVC, IDE Netbeans, iReport and MySql Server.It is recommended the learning and the use of framework Spring MVC because at the moment they are of great utility of reduce the time of development.

BIBLIOGRAFIA DE INTERNET

1. **BLOG AQUA.IT, COMO ELEGIR UN FRAMEWORK WEB**

   http://blog.aquait.info/2009/09/como-elegir-un-framework-web/

   2012-02-01

2. **BLOG JAIME CARMONA LOECHES, INTRODUCCIÓN A SPRING MVC,**

   http://jaimecarmonaloeches.blogspot.com/2012/01/introduccion-spring-mvc.html

   2012-03-12

3. **BLOG JOEL, COMPRENDIENDO STRUTS2 [PARTE 1]**

   http://joeljil.wordpress.com/2010/05/31/struts2/

   2012-07-25

4. **BLOG JOSÉ ARRARTE, MANEJO TRANSACCIONAL DE LA BASE DE DATOS CON SPRING FRAMEWORK Y AOP**

   http://josearrarte.com/blog/2009/08/12/manejo-transaccional- de- la-base-de-datos-con-spring-framework-y-aop/

   2012-05-16

5. **BLOG JOSÉ ANTONIO SAIZ, INTRODUCCIÓN A APLICACIONES REST CON SPRING 3.0 WEB MVC FRAMEWORK**

   http://joseantoniosaiz.es/blog/introduccion-aplicaciones-rest-con-spring-3-0-web-mvc-framework/

   2012-03-18

6. **BLOG HAY DIOS LIBRE, PRIMER PASO COMPLETO CON SPRING MVC 3.0 Y NETBEANS**

http://haydioslibre.blogspot.com/2010/07/spring-mvc-es-un-framework-mvc.html
2012-06-09

7. **BLOG NOSOLOLINUX, FRAMEWORKS MODELO VISTA CONTROLADOR (MVC)**
   http://www.nosolounix.com/2010/03/frameworks-modelo-vista- controlador-mvc.html
   2012-02-17

8. **BLOG VIRALPATEL.COM, SPRING 3 MVC – INTRODUCTION TO SPRING 3 MVC FRAMEWORK**
   http://viralpatel.net/blogs/tutorial-spring-3-mvc-introduction-spring-mvc-framework/
   2012-04-27

9. **CIRIANO SIERRA, JESÚS, DISEÑO E IMPLEMENTACIÓN DE UN FRAMEWORK DE PRESENTACIÓN PARA J2EE**
   http://openaccess.uoc.edu/webapps/o2/bitstream/10609/6981/1/jcirianoTFC_memoria.pdf
   2012-04-13

10. **FORO STACKOVERFLOW, HOW TO ADD CSS TO STRUTS2 TAGS**
    http://stackoverflow.com/questions/2395070/how-to-add-css-to-struts2-tags
    2012-02-03

11. **INTRODUCCIÓN A SPRING FRAMEWORK JAVA**
    http://picandocodigo.net/2010/introduccion-a-spring-framework-java/
    2012-04-13

12. **KAOLONG, FRAMEWORKS DE CAPA DE PRESENTACIÓN**

http://www.slideshare.net/kaolong/fmk-capa-de-presentacion
2012-04-01

13. MARTÍN SIERRA, ANTONIO, STRUTS 2°

http://es.scribd.com/doc/91414601/103/BENEFICIOS-DEL-USO-DE-STRUTS-2
2012-05-01

14. OPEN SOURCE WEB FRAMEWORKS IN JAVA

http://java-source.net/open-source/web-frameworks
2012-03-20

15. PATRONES DE DISEÑO EN APLICACIONES WEB CON JAVA J2EE

http://java.ciberaula.com/articulo/diseno_patrones_j2ee/
2012-04-10

16. RAMÍREZ GALLARDO, ANTONIO, CLASES Y CARACTERÍSTICAS DE J2EE

http://es.scribd.com/doc/34022342/7/Clases-caracteristicas-de-J2EE
2012-02-27

17. SITE MIGUEL TRIANA, RESUMEN SPRING

http://www.emtg.net78.net/2012/04/08/spring_summary.html
2012-05-10

18. SPRING WEB MVC

http://www.sicuma.uma.es/sicuma/independientes/argentina08/Badaracco/mvc.htm
2012-04-22

19. **TUTORIALES DE PROGRAMACIÓN EN JAVA, STRUTS2 – PARTE 1: CONFIGURACIÓN**

http://www.javatutoriales.com/2011/06/struts-2-parte-1-configuracion.html

2012-05-12

20. **TUTORIAL DE JAVASERVER FACES**

http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf

2012-02-27

21. **TUTORIAL, DECODIGO.COM, EJEMPLO DE STRUTS2, EL HOLA MUNDO**

http://www.decodigo.com/2009/06/6-ejemplo-struts2_17.html

2012-05-01

22. **Wiki, Apache Tapestry, 6 de septiembre de 2010**

http://wikis.uca.es/wikiii/index.php/Apache_Tapestry

2012-03-20

Anexos

Anexo 1: *Implementación del módulo de prueba con Struts2*

Anexo 2: *Implementación del módulo de prueba con Spring MVC*

Anexo 3: *SRS*

ANEXO 1

Implementación del módulo de prueba con Struts2

1. Instalación de Strust2.

El primer paso para instalar Struts2 es tener instalado y levanto un servidor Apache, Mysql y descargarse el plugin framework y de iReports del siguiente link:

http://www.mediafire.com/?mmwibb59uph75

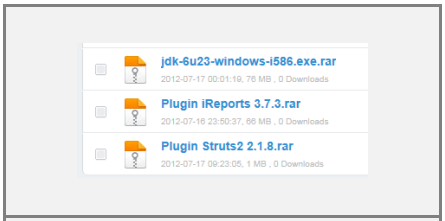


Figura 1: Descarga de Plugins

Posteriormente se procede a ejecutar Netbeans IDE 7.1.2 en donde se selecciona Tools de la barra de herramientas para añadir los Plugis necesitados.

En la pestaña Downloaded de la ventana de Plugins se selecciona los plugins tanto de Struts2 como los de iReport para proceder a instalarlos.

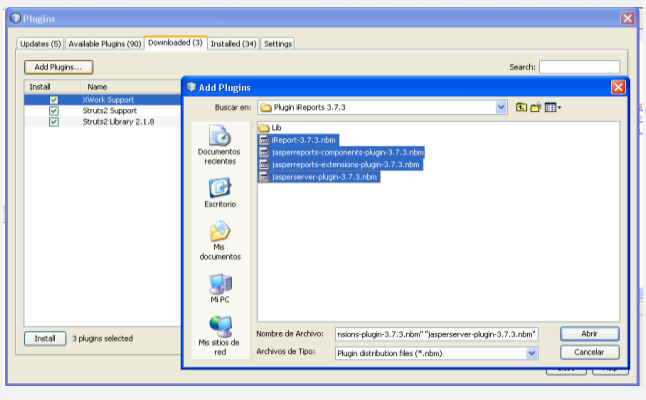


Figura 2: Añadiendo Plugins a instalar

Una vez añadidos a la lista se instala los plugins. Terminado este proceso se envia a reiniciar el IDE.

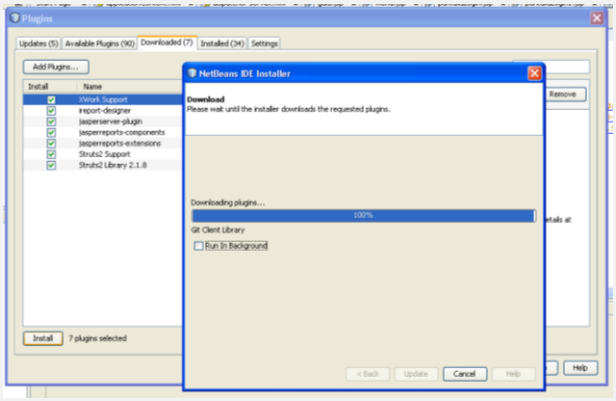


Figura 3: Instalación de Plugins

Se crea una nueva aplicación web en java y cuando se llega el momento de seleccionar el framework a usar ya se encuentra Struts2 que se ha instalado previamente.

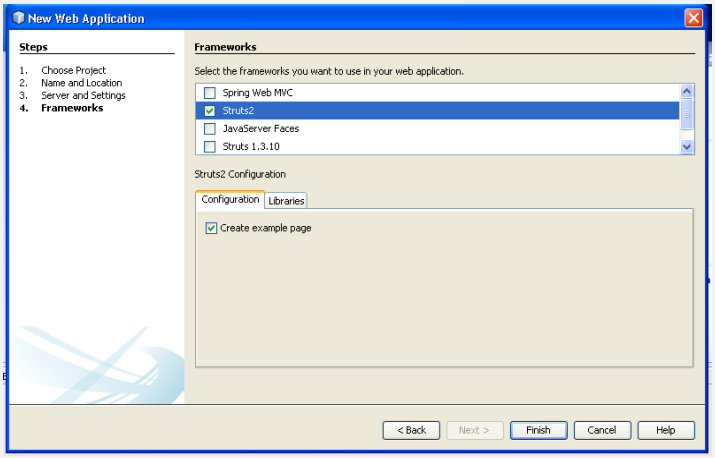


Figura 4: Añadiendo Strust2 al proyecto

Y de esta manera se cuenta con Struts2 en el proyecto

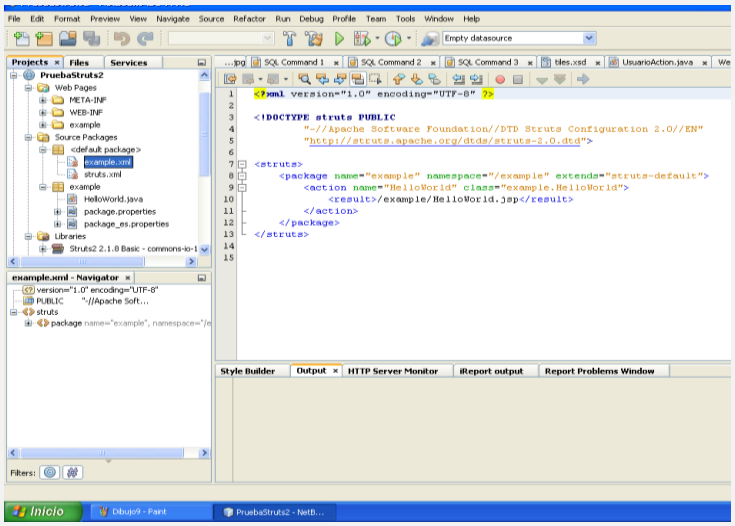


Figura 5: Estructura del proyecto con Struts2

2. Trabajando con Struts2

2.1. Trabajo con el modelo de datos

El siguiente paso para desarrollar el modulo de prueba es el manejo de la base de datos la misma que se realizo en MySQL, para a conexión se lo realizo con el JDBC.

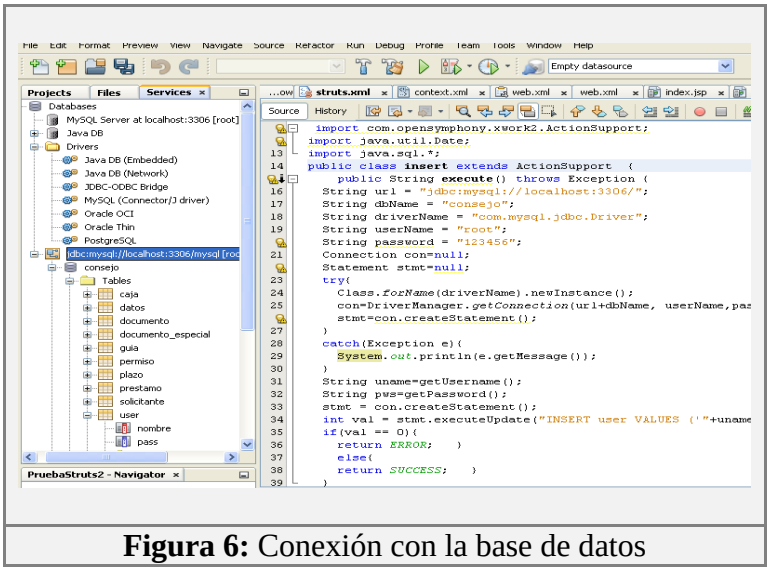


Figura 6: Conexión con la base de datos

En la aplicación se va a necesitar definir ciertos comportamientos de los objetos como:

- ✓ En la clase UsuarioMySql se encuentra el siguiente método que permite almacenar un nuevo registro.

```
public int registraUsuario(Usuario objUsuario) throws Exception{
int resultado =0;
this.conex = DriverManager.getConnection(CadenaConexion.Cadena,
CadenaConexion.usuario, CadenaConexion.contrasena);
String sql = "insert into
consejo.usuario(ci,nombres,apellidos,clave")values (?,?,?,?,?,?,?)";
PreparedStatement stmt = conex.prepareStatement(sql);
stmt.setDouble(1, objUsuario.getci());
stmt.setString(2, objUsuario.getnombres());
stmt.setString(3, objUsuario.apellidos());
stmt.setString(4, objUsuario.getclave());
resultado = stmt.executeUpdate();
return resultado;
}
```

- ✓ Cuando se desea implementar consultas a la base de datos, en este caso se utiliza UsuarioMySql para escribir un método que devuelva todos los usuarios que posee el sistema.

```
public List<Usuario> listaUsuariosReporte() throws Exception {
ArrayList<Usuario> usuarios= new ArrayList<Usuario>();
String consulta="select * from consejo.usuario";
this.conex = DriverManager.getConnection(CadenaConexion.Cadena,
CadenaConexion.usuario, CadenaConexion.contrasena);
this.stmt=conex.createStatement();
this.rs=stmt.executeQuery(consulta);
while(rs.next())
  {
   objSe.setCi(rs.getString(1));
   objSe.setNombres(rs.getString(2));
   objSe.setApellidos(rs.getString(3));
   objSe.setClave(rs.getString(4));
   usuarios.add(objSe);
   }
  return usuarios;
}
```

- ✓ Se implementa un método en la misma clase que realice una consulta a la base de datos para que un usuario del sistema se pueda loguear.

```
public Usuario ConsultarSesion(String nom) throws Exception {
    String consulta="select * from consejo.usuario where ((ci like
'"+nom+"'))";
this.conex = DriverManager.getConnection(CadenaConexion.Cadena,
CadenaConexion.usuario, CadenaConexion.contrasena);
this.stmt=conex.createStatement();
this.rs=stmt.executeQuery(consulta);
while(rs.next())
  {
   objSe.setCi(rs.getString(1));
   objSe.setNombres(rs.getString(2));
   objSe.setApellidos(rs.getString(3));
   objSe.setClave(rs.getString(4));
   usuarios.add(objSe);
   }
return usuarios;
}
```

2.2. Trabajo con la vista.

La vista es con lo que interactúa el usuario final y esta debe servir únicamente para enviar y mostrar información.

- ✓ La pagina RegistrarUsuario. jsp permite almacenar un nuevo registro la misma que contendrá el siguiente código.

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<body>
<h2> Registro de Usuarios al sistema </h2>
<s:form action="RegistraUsuario" method="post">
<table>
<tr>
<td> <s:textfield label="Ci" name="usuario.ci" /> </td>
</tr>
<tr>
<td> <s:textfield label="Nombres" name="usuario.nombres" /> </td>
</tr>
<tr>
<td> <s:textfield label="Apellidos" name="usuario.apellidos" />
</td>
</tr>
```

```
<tr>
<td> <s:textfield label="Clave" name="usuario.clave" /> </td>
</tr>
<tr>
<td> <s:password label="Password" name="usuario.clave" /> </td>
</tr>
<tr>
<td align="right" >
<s:submit name="registra" value="Registrar" type="submit" />
</td>
</tr>
</table>
</s:form>
</body>
</html>
```

Se registrar el action "RegistraUsuario" en el archivo struts.xml

```
<action name=" RegistraUsuario " class="ActionJava.UsuarioAction"
method="registra" >
<result name="exito" >/ IngresoSatisfactorio.jsp </result>
</action>
```

- ✓ La pagina ReportesrUsuario. jsp permite enlazar al reporte el mismo que contendrá el siguiente código.

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<body>
<h2> Reportes de Usuarios </h2>
<s:form action="reporteUsuarios">
<table>
<tr>
<td align="right" >
<s:submit name="lista" value="Mostrar" type="submit" />
</td>
</tr>
</table>
</s:form>
</body>
</html>
```

Registrar el action "reporteUsuarios" en el archivo struts.xml

```
<action name="reporteUsuarios" class="ActionJava.UsuarioAction"
method="listadeUsuarios" >
<result name="exito" type="jasper">
<param name="location">/reportes/reporteUsuarios.jasper</param>
<param name="dataSource">listaUsuarios</param>
<param name="contentDisposition">
attachment;filename="contacts.pdf"
</param>
<param name="format">PDF</param>
```

- ✓ La pagina logueo. jsp que permite a un usuario loguearse en el sistema, el mismo que contendrá el siguiente código.

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<body>
   <h2> Bienvenidos </h2>
<s:form action="logueo" method="post">
<table>
<tr>
<td> <s:textfield label="Ci" name="usuario.ci" /> </td>
</tr>
<tr>
<tr>
<td> <s:password label="Password" name="usuario.clave" /> </td>
</tr>
<tr>
<td colspan="2" align="right" >
<s:submit name="loguea" value="Aceptar" type="submit" />
</td>
</tr>
</table>
</s:form>
```

- ✓ Registrar el action "logueo" en el archivo struts.xml

```
<package name="default" namespace="/" extends="struts-default">
<action name="logueo" class="ActionJava.UsuariooAction" >
<result name="error" >/logueo.jsp </result>
<result name="exito" >/bienvenida.jsp </result>
</action>
</package>
```

2.3. Trabajo con el controlador

Las acciones son el corazón de la aplicación, puesto que contienen toda la lógica de la aplicación. Las acciones utilizan el modelo y definen variables para la vista. Consiste en un archivo en el que se encuentra definidas varias funciones, una por cada vista.

- ✓ El código del Controlador necesario para realizar la tarea de registrar a los usuarios se encuentra en la clase UsuarioAction la misma que debe contar con el siguiente método:

```
public String registra(){
Usuario user;
UserService servicioUsuario;
String vista="exito";
try {
servicioUsuario.registrarUsuario(user);
cliente.setNombre("");
} catch (Exception e) {
e.printStackTrace();
}
return vista;
}
```

- ✓ El código del Controlador para realizar la tarea de listar los usuarios es en la clase UsuarioAction el mismo que contendrá el método listadeUsuarios.

```
private List<Usuario> listaUsuarios;
public List<Usuario> getListaUsuarios() {
return listaUsuarios;
}
public void setListaUsuarios(List<Usuario> listaUsuarios) {
this.listaUsuarios = listaUsuarios;
}

public String listadeUsuarios(){
String vista="exito";
  UserService servicioUsuario;
try {
this.setListaUsuarios(servicioUsuario.listarUsuariosReporte());
} catch (Exception e) {
e.printStackTrace();
}
return vista;
}
```

- El código del Controlador para realizar la tarea de listar los usuarios es en la clase UsuarioAction el mismo que contendrá el método listadeUsuarios.

```
public String execute() throws Exception{
 String vista="error";
 UsuarioService LS = new UsuarioService();
 Usuario usuarioCandidato= new Usuario ();
 usuarioCandidato.setCi(this.getCi());
 Usuario objUsuario=null;
 try {
 objUsuario = LS.validaUser(usuarioCandidato);
 }
 catch (Exception e) {
   e.printStackTrace();
   }
   if(objUsuario!=null){
   if(objUsuario.getClave().equals(this.getClave())){
   Map <String,Object> lasesion=
ActionContext.getContext().getSession();
   lasesion.put("b_usuario", objUsuario);
   vista="exito";
   }
   }
  return vista;
   }
```

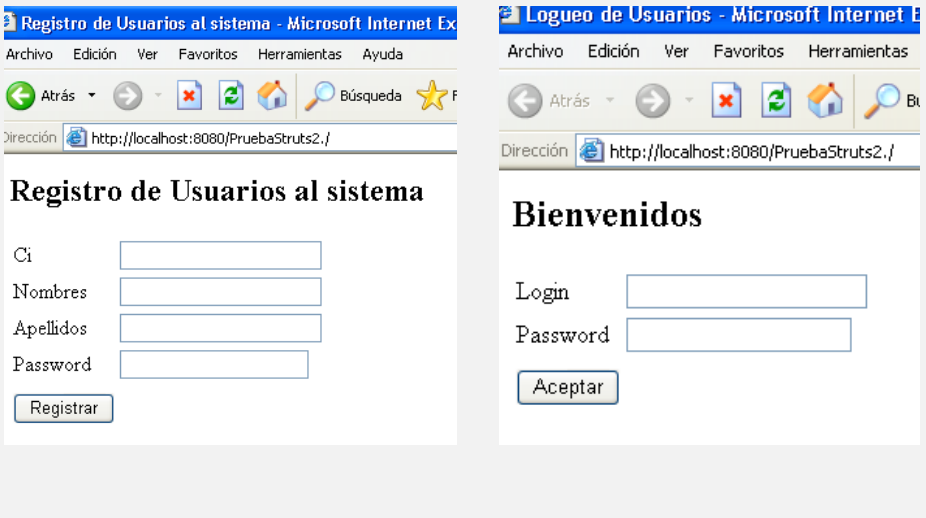


Figura 7: Pantallas de registro y logueo de usuario

ANEXO 2

Implementación del módulo de prueba con Spring MVC

1. Creación del modulo de prueba con Spring MVC

Para la creación del modulo de prueba se debe instalar previamente:

- ✓ Herramienta Java (JDK)
- ✓ IDE Netbeans 7.1
- ✓ Apache Tomcat
- ✓ MySql

En el IDE netbeans 7.1 viene incorporado el framework Spring MVC versión 3. Se crea una nueva aplicación web en el paso cuatro se selecciona el framework Spring Web MVC

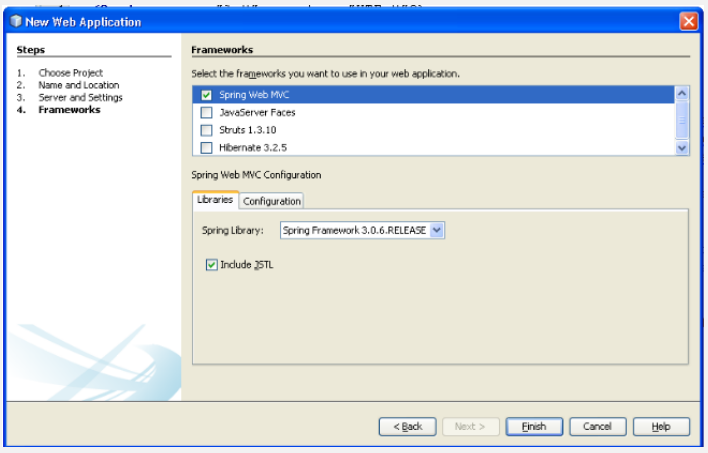


Figura 1: Añadiendo Spring MVC al proyecto

La figura a continuación muestra como se vería el proyecto con las librerías de SpringMVC

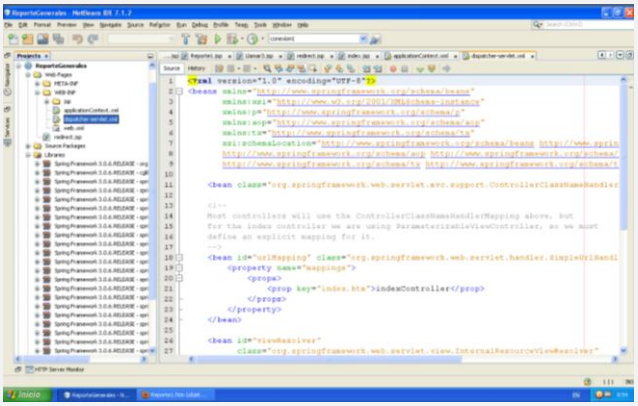


Figura 2: Estructura del proyecto con Spring MVC

2. Trabajando con Spring MVC

El siguiente paso para desarrollar el modulo de prueba es el manejo de la base de datos la misma que se realizo en MySQL, para la conexión se lo realizo con el JDBC.

Se empieza creando las clases que son:

- ✓ Los beans para la comunicación vista-control, se llamará BeanVistaControl.
- ✓ Los servicios, se llamará Servicios.
- ✓ Los controladores, se llamará Controladores.

El controlador maneja las vistas, los modelos y los servicios. El bean es un POJO que contiene la información a mostrar en la Vista. Solo tiene getters&setters (generalmente). Los servicios, contienen la llamada "lógica de negocio".

Dentro de Source Package, en el BeanVistaControl, se crea IngresoSistema.java

```
package BeanVistaControl;

import java.sql.Date;
import java.sql.Time;

public class IngresoSistema
{
 private int IdTabla;
    private int IdUsuario;
    private Date fecha;
    private Time hora;


    public int getIdTabla() {
        return IdTabla;
    }

    public void setIdTabla(int IdTabla) {
        this.IdTabla = IdTabla;
    }

    public int getIdUsuario() {
        return IdUsuario;
    }

    public void setIdUsuario(int IdUsuario) {
        this.IdUsuario = IdUsuario;
    }

    public Date getFecha()
    {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    public Time getHora() {
        return hora;
    }


    public void setHora(Time hora) {
        this.hora = hora;
    }
    public String toString()
        {
                return ""+getIdTabla()+" "+getIdUsuario()+" "+getFecha()+" "+getHora();
        }
}
```

En el BeanVistaControl, se crea también Usuario.java

```
package BeanVistaControl;

public class Usuario
{
    private int IdUsuario;
    private String ci;
    private String nombres;
    private String apellidos;
    private String clave;

    public int getIdUsuario() {
        return IdUsuario;
    }

    public void setIdUsuario(int IdUsuario) {
        this.IdUsuario = IdUsuario;
    }

    public String getCi() {
        return ci;
    }

    public void setCi(String ci) {
        this.ci = ci;
    }

    public String getNombres() {
        return nombres;
    }

    public void setNombres(String nombres) {
        this.nombres = nombres;
    }

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public String getClave() {
        return clave;
    }

    public void setClave(String clave) {
        this.clave = clave;
    }
}
```

Dentro de Source Package, en Servicios, se crea Ingreso.java

```
package Servicios;

import BeanVistaControl.IngresoSistema;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

public class Ingreso
{
public Collection principal(IngresoSistema ing) {
        // Datos de conexion con la base de datos
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName("org.gjt.mm.mysql.Driver");
        dataSource.setUrl("jdbc:mysql://localhost/consejo2");
        dataSource.setUsername("root");
        dataSource.setPassword("1234567");

        // La clase Spring con la Connection
        JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

        // Realizamos la consulta
        Collection personas = jdbcTemplate.query(
                "select * from ingresosistema ", new RowMapper() {

                    @Override
                    public Object mapRow(ResultSet rs, int arg1)
                            throws SQLException {
                        // Se rellena un bean Persona a partir de la fila actual
                        // del ResultSet
                        IngresoSistema persona = new IngresoSistema();
                        persona.setIdUsuario(rs.getInt("IdUsuario"));
                        persona.setIdTabla(rs.getInt("IdTabla"));
                        persona.setFecha(rs.getDate("fecha"));
                        persona.setHora(rs.getTime("hora"));
                        return persona;
                    }
                });

        // Escribimos en pantalla los datos leidos
        /*for (Object persona : personas) {
            System.out.println(persona.toString());
        }*/
        return personas;
    }

}
```

En Servicios, se crea LoguearUsuario.java

```
package Servicios;

import BeanVistaControl.Usuario;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowCallbackHandler;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

public class loguearUsuario
{
    Usuario persona = new Usuario();
    public DriverManagerDataSource conexion()
    {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName("org.gjt.mm.mysql.Driver");
        dataSource.setUrl("jdbc:mysql://localhost/consejo2");
        dataSource.setUsername("root");
        dataSource.setPassword("123456");
        return dataSource;
    }
     // La clase Spring con la Connection
         // Datos de conexion con la base de datos

     public String login (Usuario usr)
    {
        try
        {
        // La clase Spring con la Connection
        JdbcTemplate jdbcTemplate = new JdbcTemplate(conexion());

        // Realizacmos la consulta
        jdbcTemplate.execute("insert into usuario (ci,nombres,apellidos,clave) values
('"+usr.getCi()+"','"+usr.getNombres()+"','"+usr.getApellidos()+"','"+usr.getClave()+
"')");
        return "Usuario Insertado satisfactoriamente";
        }
        catch(Exception ex)
                {
                    return "error";
                }
    }
        }
```

Dentro de Source Package, en Controladores, se crea IngresoSisController.java

En Spring se tipos de diferentes controllers se ha utilizado SimpleFormController que es usado para desplegar y procesar un formulario, bajo el mismo componente.

```
package controladores;

import BeanVistaControl.IngresoSistema;
import Servicios.Ingreso;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.SimpleFormController;

public class IngresoSisController extends SimpleFormController {

    public IngresoSisController() {
        setCommandClass(IngresoSistema.class);
        setCommandName("beanIngreso");
        setSuccessView("PantallaReporte");
        setFormView("Reporte");
    }

    public Ingreso getIngreso() {
        return Ingreso;
    }

    public void setIngreso(Ingreso Ingreso) {
        this.Ingreso = Ingreso;
    }
     private Ingreso Ingreso;


     @Override
    protected ModelAndView onSubmit(Object command) throws Exception
    {
        IngresoSistema name=(IngresoSistema)command;
        ModelAndView mv = new  ModelAndView(getSuccessView());
        mv.addObject("mensajeLogin",name.getIdTabla());
        return mv;
    }
}
```

En Controladores, se crea LoginController.java

```
import BeanVistaControl.Usuario;
import Servicios.loguearUsuario;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.SimpleFormController;

public class LoginController extends SimpleFormController {

    public LoginController()
    {
        setCommandClass(Usuario.class);
        setCommandName("beanUsuario");
        setFormView("pantallaLogin");

        setSuccessView("pantallaPrincipal");
    }
    private loguearUsuario loguearUsuario;

    public loguearUsuario getLoguearUsuario() {
        return loguearUsuario;
    }

    public void setLoguearUsuario(loguearUsuario loguearUsuario) {
        this.loguearUsuario = loguearUsuario;
    }

  @Override
    protected ModelAndView onSubmit(Object command) throws Exception
    {
        Usuario name=(Usuario)command;
        ModelAndView mv = new  ModelAndView(getSuccessView());
        mv.addObject("mensajeLogin",loguearUsuario.login(name));
        return mv;
    }
}
```

Después se trabaja en el WEB-INF primeramente en el applicationContext.xml agregando el bean que se muestra en la siguiente figura:

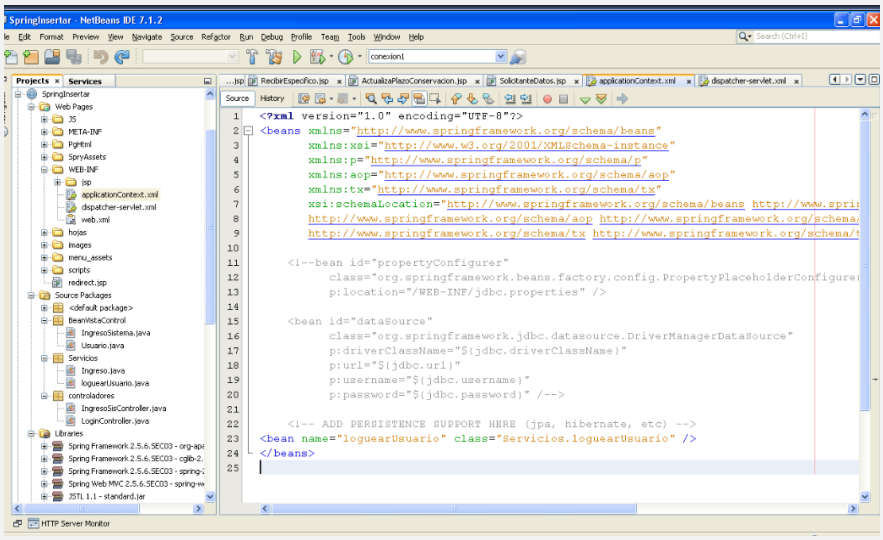


Figura 3: applicationContext.xml

Luego en el dispatcher-servlet.xml

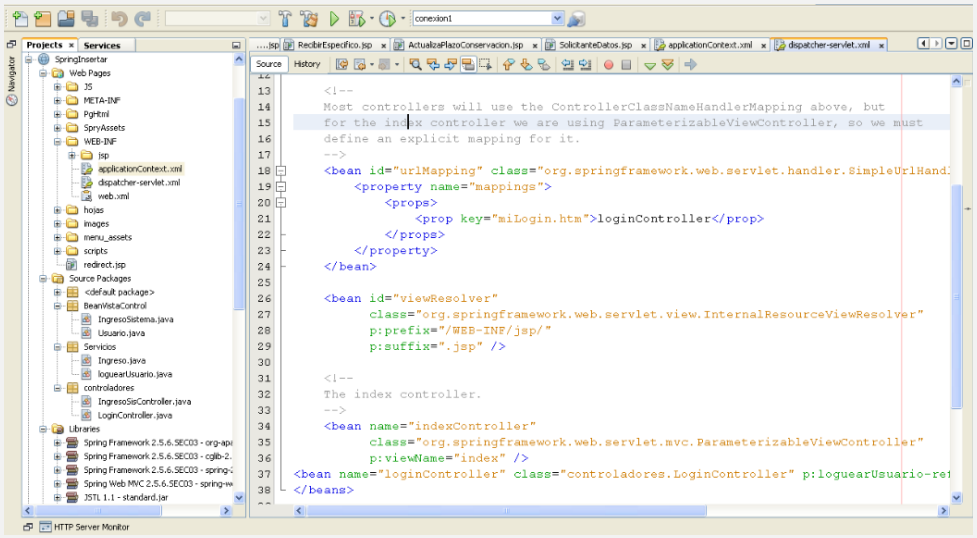


Figura 4: dispatcher-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xmlns:p="http://www.springframework.org/schema/p"

       xmlns:aop="http://www.springframework.org/schema/aop"

       xmlns:tx="http://www.springframework.org/schema/tx"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

       http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd

       http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">



    <bean
class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"
/>

    <bean id="urlMapping"
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">

        <property name="mappings">
```

Dentro del dispatcher-servlet.xml está el Handler Mapping que tiene como objetivo de indicarle cual sera el componente que debe recibir el request enviado por el usuario.

Por lo cual el dispatcher-servlet.xml, le pregunta a uno o más Handler Mapping cual será le Controller que recibirá el Request. El Handler Mapping utilizado es SimpleUrlHandlerMapping que mapea controladores a URL basándose en una colección de propiedades que se definen en el Spring applicationContext.xml

El dispatcher-servlet.xml delega la responsabilidad de la mapping del nombre logico de la vista, con el componente a utilizar al ViewResolver. El ViewResolver es el encargado de realizar el mapping entre el nombre lógico de la vista y el componente. El ViewResolver

utilizado es el InternalResourceViewResolver que resuelve el nombre lógico utilizando el mapping a velocity y JSP.

Como resultado se obtiene:

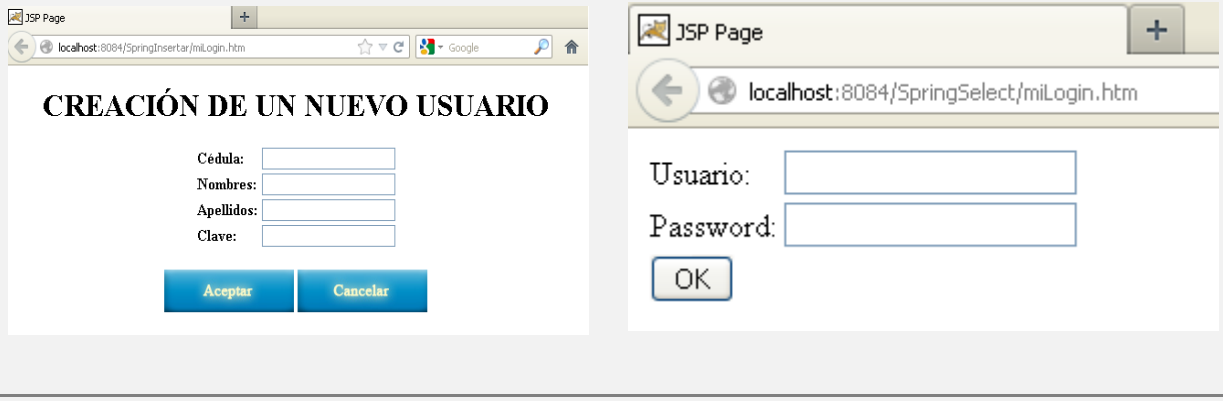


Figura 5: Pantalla de creación y logueo de usuario